PER
TI99/4A - VIC20 - C64
ZX81 - ZX SPECTRUM - MZ700
IBM PC - M20 - APPLE

Martino Sangiorgio

il manuale del basic

PER
TI99/4A - VIC20 - C64
ZX81 - ZX SPECTRUM - MZ700
IBM PC - M20 - APPLE

Martino Sangiorgio

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

SOMMARIO

INTRODUZIONE

SCOPO DEL MANUALE

SINTASSI DI UN LINGUAGGIO

FORMALISMI SINTATTICI

BNF
CARTE SINTATTICHE
SINTASSI DI TIPO COBOL

DIALETTI DEL BASIC CONSIDERATI

SCOPO DEL MANUALE

Data la vitalità che ha scosso, a partire dal Natale 1983 e per tutto l'84, il mercato degli HOME e PERSONAL Computer in Italia, si è venuta a creare una notevole confusione di terminologia, per lo più riguardante il BASIC, tra i possessori di elaboratori. Non sono pochi infatti coloro che sono convinti che il BASIC sia un linguaggio universale per cui, dopo aver imparato ad usare, per esempio, un VIC 20, si possa tranquillamente passare, senza alcun problema, a programmare uno SPECTRUM. Ora, questo è vero solo in parte. Se infatti la struttura fondamentale del BASIC è comune e le istruzioni di carattere generale sono normalmente implementate, con poche varianti di tipo sintattico, su quasi tutti i microelaboratori attuali, è anche vero che alcuni gruppi di istruzioni (riguardanti, per esempio, la grafica) sono notevolmente diversi da un elaboratore all'altro. Si tende a condensare tutto questo dicendo che, su ogni elaboratore, è presente un particolare "dialetto" del BASIC. Lo scopo di questo libro è dunque quello di confrontare tutte le "parole", le "frasi", di alcuni di questi dialetti, in modo da fornire al lettore una panoramica il più possibile completa delle istruzioni BASIC dei microcalcolatori attualmente più diffusi, fornendo, nel contempo, la chiave per poter procedere, in un momento successivo, alla conversione di programmi da un elaboratore ad un altro.

Il primo passo da compiere è ora quello di sapere cosa si intende per "sintassi" di un linguaggio e di conoscere il formalismo sintattico utilizzato nel libro.

SINTASSI DI UN LINGUAGGIO

Nella definizione di un linguaggio, e non solo di un linguaggio di programmazione, si scorgono tre momenti fondamentali:

- LA DEFINIZIONE DELLA SINTASSI
- LE DISCUSSIONI SULLA SEMANTICA
- LA DEFINIZIONE DELLE FORME GRAMMATICALI.

La *sintassi* ci mostra come costruire o riconoscere le frasi corrette del linguaggio.
La *semantica* dà significato ad ogni elemento della sintassi costruita.
La *grammatica* dà delle regole sulla composizione di elementi del linguaggio per formare elementi più evoluti.
Nei linguaggi di programmazione, causa le regole relativamente semplici di composizione degli elementi, il ruolo della grammatica viene normalmente demandato alla sintassi.

FORMALISMI SINTATTICI

Esistono diversi modi di formalizzare la sintassi di un linguaggio di programmazione (linguaggio di definizione) in termini di un linguaggio parlato (es. l'italiano, linguaggio di riferimento). Vediamone alcuni.

BNF (Backus-Naur Form)

Esempio:

<comando LIST> : : =LIST | LIST <numero linea> |

LIST <numero linea> — |
LIST — <numero linea> |

LIST <numero linea> — <numero linea>

In questa notazione le parentesi " < >" racchiudono il *nome* (in linguaggio di riferimento) di un oggetto o di una classe di oggetti del linguaggio di definizione. Il simbolo ": : =" ha il significato di "è costituito da" ed il simbolo " | " ha il significato di "oppure", indica cioè una alternativa.

Nell'esempio citato i simboli <comando LIST> e <numero linea>, sono simboli metalinguistici, o metasimboli, o simboli *non terminali,* per il fatto che sono ulteriormente definibili.

Infatti, ad esempio:

<numero linea> : : = <cifra> | <numero linea> <cifra>

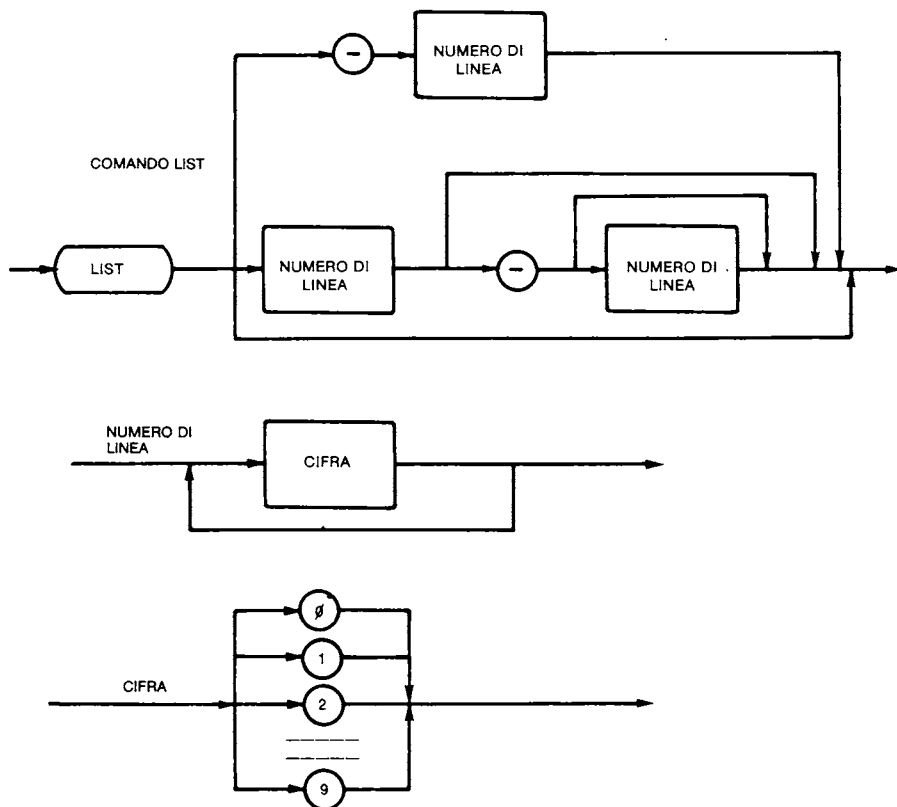
indica che un numero di linea è costituito o da una cifra oppure da un numero di linea seguito da una cifra (esempio di ricorsività), mentre:

<cifra> : : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

indica che una cifra è costituita o dal numero 0, o dal numero 1, ecc. I simboli 0, 1, 2 ecc. che si trovano alla destra del connettore ": : =" sono simboli *terminali*, perchè non sono ulteriormente definibili. Così pure per i simboli LIST e — dell'esempio iniziale.

CARTE SINTATTICHE

Un formalismo sintattico, introdotto nella definizione del Pascal, ma molto interessante e utile dal punto di vista pratico, è costituito dalle *carte sintattiche*. In tale formalismo i simboli terminali vengono rappresentati rinchiusi in cerchi (od ellissi), mentre i simboli non terminali vengono rappresentati da elementi del linguaggio di riferimento (es. l'italiano), rinchiusi in rettangoli.

In tale formalismo gli esempi visti in precedenza diventano:

A causa della maggior chiarezza e immediatezza di tale formalismo rispetto agli altri, lo abbiamo adottato (seguendo l'esempio della Olivetti per il suo M20) per formalizzare le istruzioni e i comandi BASIC.

SINTASSI DI TIPO COBOL

È forse il formalismo più utilizzato per definire i comandi e le istruzioni BASIC per i piccoli calcolatori. È stato utilizzato per descrivere la sintassi del COBOL. In esso si usano parentesi graffe ("{ }") per descrivere alternative di simboli, parentesi quadre ("[ ]") per descrivere elementi opzionali, una serie di punti ("....") per indicare elementi ricorsivi. Gli elementi visti in precedenza, con questo formalismo diventano:

comando LIST : : = LIST [num. riga] [— [num. riga]]

num. riga : : = cifra [[cifra] ......]

$$\text{cifra} ::= \begin{Bmatrix} 0 \\ 1 \\ 2 \\ \\ \dots \\ \dots \\ 9 \end{Bmatrix}$$

In sostanza la sintassi permette di:

— generate *tutti* (e solo) gli elementi del linguaggio che essa definisce

— riconoscere se un certo elemento appartiene o no al linguaggio che essa definisce.

DIALETTI DEL BASIC CONSIDERATI

Delle dozzine di dialetti BASIC che ormai hanno invaso il mondo, abbiamo scelto quelli dei nove elaboratori che più si sono affermati nel nostro Paese. Eccone l'elenco completo, che comprende anche la versione presa in esame:

1) TEXAS TI 99/4A: TI - BASIC

2) COMMODORE VIC 20: BASIC per il CBM VIC 20

3) COMMODORE 64: BASIC per il CBM 64

4) SINCLAIR ZX 81: BASIC

5) SINCLAIR ZX Spectrum: BASIC

6) SHARP MZ-700 (711-721-731): BASIC 1Z-013B

7) IBM PC: IBM BASIC per il Personal Computer IBM, versione 1 (avanzato)

8) OLIVETTI M20: BASIC Microsoft per M20

9) APPLE: BASIC Applesoft per APPLE II

Di tutti questi elaboratori abbiamo preso in considerazione solamente l'interprete BASIC che viene consegnato al cliente all'atto dell'acquisto dell'elaboratore. Non abbiamo quindi considerato, ad esempio, il TI-EXENDED BASIC per il TEXAS, il SIMON BASIC per il CBM 64, il DISK BASIC per lo SHARP e tutti i comandi del PCOS e del DOS che possono essere richiamati dal BASIC, rispettivamente, di M20 e di APPLE. Questo sia per ovvi motivi di spazio sia perchè siamo convinti che l'introduzione delle istruzioni implementate in tutti questi interpreti, avrebbe aumentato, anzichè limitarla, la confusione che già regna.

CAPITOLO 1

OPERATORI, COSTANTI E VARIABILI - MATRICI

Prima di procedere, elaboratore per elaboratore, alla classificazione degli operatori, delle costanti, delle variabili, vediamo cosa dice a proposito il BASIC Standard ANSI - BSRX 3.60 del 1978; confronteremo poi le nove versioni con questa.

OPERATORI

Aritmetici

— Elevamento a potenza	^
— Cambio di segno	—
— Moltiplicazione	*
— Divisione	/
— Addizione	+
— Sottrazione	—

Relazionali

— Uguaglianza	=
— Disuguaglianza	<> o ><
— Minore	<
— Maggiore	>
— Minore o uguale	<= o =<
— Maggiore o uguale	>= o =>

TABELLA DELLE COSTANTI E DELLE VARIABILI NUMERICHE					
COSTANTI NUMERICHE	**SIMBOLO**	**RANGE**	**BYTE**	**CIFRE SIGNIFICATIVE**	**CIFRE VISUALIZZATE**
Intere	%	—32767 ÷ + 32767	2	5	5
Reali in virgola fissa					
● singola precisione	!	$\pm 10^{-38} \div \pm 10^{+38}$	4	7	6
● doppia precisione	#	$\pm 10^{-3o8} \div \pm 10^{+3o8}$	8	16	15
Reali in virgola mobile					
● singola precisione	E	$\pm 10^{-38} \div \pm 10^{+38}$	4	7	6
● doppia precisione	D	$\pm 10^{-3o8} \div \pm 10^{+3o8}$	8	16	15
Ottali	&	0 ÷ 177777	2		
Esadecimali	&H	0 ÷ FFFF	2		
VARIABILI NUMERICHE					
Intere	%	—32767 ÷ + 32767	2	5	5
Reali					
● singola precisione	!	$\pm 10^{-38} \div \pm 10^{+38}$	4	7	6
● doppia precisione	#	$\pm 10^{-3o8} \div \pm 10^{+3o8}$	8	16	15

COSTANTI ALFANUMERICHE (O STRINGA)

Devono essere racchiuse tra doppi apici (")

Esempi:

"NOME, 1,2,3"

"PIPPO"

Ecc.

VARIABILI ALFANUMERICHE (O STRINGA)

Devono terminare col segno del dollaro ($)

Esempi:

NOME $

PIPPO $ = "PIPPO"

OPERATORI SU STRINGHE

— Concatenamento (+)
— Operatori relazionali
— Estrazione sottostringhe (vedere "operatori funzionali").

NOMI DI VARIABILI

I nomi delle variabili, sia numeriche che alfanumeriche, possono essere di soli due caratteri nella versione più limitata del BASIC, mentre possono essere lunghi fino a 40 caratteri nelle versioni più estese.

Nella versione a due caratteri, immettendo un nome più lungo non si ha segnalazione di errore, ma solo i primi due vengono presi in considerazione. Quindi, ad esempio, le variabili:

MESE e MEDIA

vengono in tal caso considerate la stessa variabile (ME).

In entrambe le versioni di BASIC, il primo carattere di definizione della variabile *deve* essere una lettera dell'alfabeto (A ÷ W), mentre per i successivi caratteri possono essere utilizzati sia lettere che numeri che alcuni eventuali simboli speciali.

Alcuni nomi non possono essere utilizzati per definire variabili; è l'esempio delle istruzioni BASIC (es. PRINT, LIST ecc.). Sono "parole riservate" all'interprete BASIC.

CONFRONTO CON GLI ELABORATORI TRATTATI

TEXAS TI 99/4A

OPERATORI

Sia gli operatori aritmetici che quelli relazionali sono gli stessi visti per lo standard.

Costanti e variabili

Non esiste suddivisione tra costante (e variabile) intera e reale, come non esistono i simboli finali per la dichiarazione del tipo; può solo essere usato il simbolo E per dichiarare le costanti nella notazione scientifica. Tutte le costanti (e variabili) vengono considerate con la stessa precisione. I valori possono essere compresi tra

$\pm\ 9.9999999999999E^{+127}$ e $\pm\ 1E^{-128}$

Non si possono definire costanti ottali o esadecimali.

Per le costanti e variabili alfanumeriche vale quanto detto per lo standard. L'operatore concatenamento non è dato da simbolo "+"; al suo posto viene usato il simbolo "&".

Il nome delle variabili può essere lungo fino a 15 caratteri (compreso il $ per le variabili alfanumeriche), tutti significativi.

COMMODORE VIC 20 e 64

OPERATORI

Sono gli stessi visti nello standard, tranne l'elevamento a potenza per il quale si usa il simbolo "↑".

Gli operatori relazionali sono gli stessi dello standard.

Costanti e variabili

Non esiste la doppia precisione, per cui gli unici simboli per la definizione del tipo di costante (o variabile) sono "%" per gli interi e "E" per la notazione scientifica. I numeri decimali (reali) non vogliono suffisso.

Non si possono definire costanti ottali o esadecimali.

I nomi delle variabili possono essere lunghi a piacere, ma solo i primi due caratteri sono significativi. Due caratteri vengono usati anche per le varibili stringa, oltre naturalmente al simbolo "$".

Esiste l'operatore concatenamento ("+") per le stringhe.

SINCLAIR ZX SPECTRUM E ZX 81

OPERATORI

Gli operatori aritmetici sono gli stessi visti per lo standard, tranne l'elevamento a potenza per il quale si usa il simbolo "↑" per lo ZX Spectrum e "**" per lo ZX 81.

Gli operatori relazionali sono gli stessi dello standard.

Costanti e variabili

Esiste solo il simbolo E per la definizione delle costanti in notazione scientifica. Tutte le costanti e le variabili vengono considerate in singola precisione. Per le costanti reali o intere possono essere usate un massimo di 9 o 10 cifre (fino al numero 4294967296). Per le costanti in notazione scientifica il "Range" è da $\pm 10^{+38}$ a $\pm 10^{-38}$

Non si possono definire costanti ottali o esadecimali.

I nomi delle variabili numeriche possono essere lunghi a piacere. I nomi delle variabili alfanumeriche devono essere di una sola lettera seguita dal simbolo "$".

Esiste l'operatore "+" (concatenamento) applicabile a stringhe.

SHARP MZ 700

OPERATORI

Gli operatori aritmetici sono gli stessi visti per lo standard, salvo l'elevazione a potenza per cui si usa il simbolo "↑".
Gli operatori relazionali sono gli stessi dello standard.

Costanti e variabili

Esiste solo il simbolo E per la definizione delle costanti in notazione scientifica. Tutte le costanti e le variabili vengono considerate in singola precisione. Il "Range" va da $\pm 10^{+38}$ a $\pm 10^{-38}$.
Si possono definire costanti esadecimali (da usarsi in concomitanza di alcune istruzioni BASIC) utilizzando come prefisso il simbolo "$". Esempio:

```
LIMIT $C000
```

I nomi delle variabili numeriche possono essere lunghi a piacere, ma solo i primi due caratteri sono significativi.
I nomi delle variabili stringa devono essere al massimo di due caratteri seguiti dal suffisso "$".
Esiste l'operatore "+" per il concatenamento di stringhe.

IBM P.C. E OLIVETTI M20

OPERATORI

Gli operatori aritmetici sono gli stessi dello standard più i seguenti:

- Divisione per intero \
- Aritmetica modulare MOD

Nella divisione per intero gli operandi vengono arrotondati agli interi prima che venga eseguita la divisione; il quoziente viene poi troncato all'intero.
L'operatore MOD fornisce il valore intero che è il resto di una divisione per intero.
Gli operatori relazionali sono gli stessi dello standard.

Costanti

Vale quanto detto nello standard.

Variabili

Vale quanto detto nello standard. In più, oltre a dichiarare le variabili attraverso i simboli finali %, !, #, $, è possibile dichiararle attraverso le specifiche BASIC:

- DEFINT definisce variabili intere
- DEFSNG definisce variabili in singola precisione
- DEFDBL definisce variabili in doppia precisione
- DEFSTR definisce variabili stringa

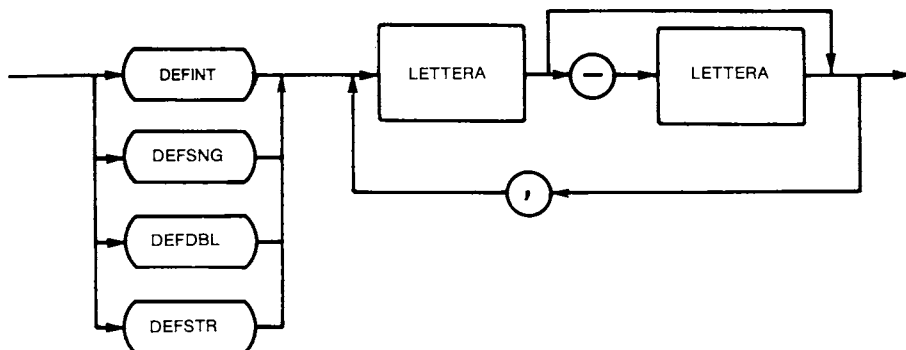
Il nome di una variabile può avere qualsiasi lunghezza, ma solo i primi 40 caratteri verranno presi in considerazione.

DEFINT/DEFSNG/DEFDBL/DEFSTR
(Istruzioni di definizione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Dichiarano il tipo di tutte le variabili il cui nome inizia con una lettera ben specificata.

Vengono normalmente inserite all'inizio del programma e devono precedere l'uso delle variabili di cui definiscono il tipo.

Formalismo sintattico:

Esempio:

```
10    DEF DBL P-R
20    DEF STR B
30    DEF INT Y, C — F
40    PERCENTUALE = 2 #/3: PRINT PERCENTUALE
50    BIBITA = "COCA COLA": PRINT BIBITA
60    Y = 20/3: PRINT Y
```

Risultato:

LINEA 10: Dichiara che tutte le variabili che iniziano con P, Q e R sono a doppia precisione.

LINEA 20: Dichiara che tutte le variabili che iniziano con B sono variabili stringa.

LINEA 30: Dichiara che tutte le variabili che iniziano con Y, C, D, E, F sono variabili intere.

LINEA 40: Effettua l'operazione specificata e stampa:

.6666666666666666

che è il risultato a doppia precisione.

LINEA 50: dà alla variabile il valore specificato e stampa:

COCA COLA.

Da notare che non è necessario, in questo caso, apporre il suffisso $, dato che le variabili inizianti con B sono state dichiarate a stringa.

LINEA 60: Effettua l'operazione e stampa:

6

perchè Y è stata definita intera.

Note:
— Queste istruzioni di dichiarazione del tipo di variabile vengono normalmente inserite all'inizio del programma, e devono comunque precedere l'uso delle variabili di cui definiscono il tipo.
— I caratteri che definiscono il tipo di variabile (esempio, % per variabili intere) hanno la precedenza sulle istruzioni di definizione: cioè un variabile definita a doppia precisione con l'istruzione DEFDBL, ma che utilizza il suffisso % (ad sempio la variabile F%), verrà considerata dal sistema come variabile intera.

APPLE

OPERATORI

Sia quelli numerici che quelli relazionali sono gli stessi visti per lo standard.

Costanti e variabili

Esistono costanti intere e reali in singola precisione, con la notazione scientifica. Esistono variabili intere (suffisso %) e reali a singola precisione (niente suffisso). Non esiste la doppia precisione. Il "range" è da $\pm 10^{+38}$ a $\pm 10^{-38}$.

I nomi delle variabili possono essere lunghi fino a 238 caratteri, ma solo i primi due sono significativi.

Per le costanti e variabili alfanumeriche vale quanto visto per lo standard.

Esiste anche l'operatore "concatenamento" (simbolo "+").

ESPRESSIONI RELAZIONALI

Una espressione di confronto paragona due espressioni numeriche o stringa mediante l'uso di operatori relazionali.

Non è possibile confrontare una espressione numerica con una espressione alfanumerica, e viceversa.

L'espressione di confronto fornisce un risultato numerico. Esso è —1 se il confronto è vero, mentre è 0 se il confronto è falso.

È possibile utilizzare espressioni relazionali nell'ambito delle istruzioni IF, ma è anche possibile utilizzarle per assegnare valori alle variabili.

Esempio:

Se A = 10, B = 5, C = 14 l'espressione:

$D = A + B > C$

assegnerà a D il valore 9, mentre l'espressione:

$D = A + B < C$

assegnerà a D il valore 10, poichè il confronto è falso.

Il confronto fra stringhe viene effettuato comparando carattere per carattere. In tal caso verrà utilizzato il valore numerico di ogni carattere, cioè il suo valore decimale dato dalla codifica ASCII.

OPERATORI LOGICI

Essi sono i seguenti nell'ordine di priorità (dal più alto al più basso), in cui essi vengono svolti dal linguaggio BASIC.

NOT	(negazione)
AND	(prodotto logico)
OR	(somma logica)
XOR	(or esclusivo)
IMP	(implicazione)
EQV	(equality o equivalenza)

ESPRESSIONI LOGICHE

Un'espressione logica è data da un certo numero di operandi separati da un operatore logico.

Gli operandi possono essere espressioni numeriche o di confronto, che forniscono un risultato numerico. Anche il risultato di un'espressione logica è numerico (intero).

Esempi:

63 AND 16: il risultato è 16. Infatti: $63_D = 111111_B$

$16_D = \underline{010000}_B$

$010000_B = 16_D$

4 OR 2: il risultato è 6. Infatti: $4_D = 100_B$

$2_D = 010_B$

$110_B = 6_D$

NOT (Operatore Logico)

	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Cambia il valore logico di una espressione logica.

Esempio:

```
10 INPUT "DAMMI LA TUA ETA'"; A
20 IF NOT (A > 5) THEN PRINT "NON È POSSIBILE"
30 INPUT R
40 IF NOT (R = —1) THEN 100
50 F% = 2
60 F% = NOT F%
100 END
```

Risultato:

LINEA 10: Stampa il messaggio evidenziato e attende la risposta dell'utente, che viene successivamente memorizzata nella variabile A.
LINEA 20: Se il contenuto della variabile A è ⩽ 5, stampa il messaggio specificato, se invece A è > 5 esegue l'istruzione seguente.
LINEA 30: attende che l'utente digiti un valore numerico, indi lo memorizza nella variabile R.
LINEA 40: Se R non è uguale a —1, il programma salta alla linea 100.
LINEA 50: Imposta la variabile intera F% a 2.
LINEA 60: Cambia tutti i valori di ogni bit della variabile F% da 0 a 1 e viceversa. Poichè F% valeva 2_D (10_B), il valore di F% dopo l'esecuzione di questa istruzione diventa:

$$F\% = 1111111111111101_B = -3_D$$

In F% sarà quindi contenuto il valore decimale —3.

Note:

— La tavola della verità dell'operatore NOT è la seguente: (V = Vero = 1; F = Falso = 0).

X	NOT X
V	F
F	V

AND (Operatore Logico)

	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Viene usato nelle espressioni logiche per legare due condizioni. L'espressione sarà vera se entrambe le condizioni sono vere, altrimenti l'espressione risulterà falsa.

Esempio:

```
10 A = 1 : B = 30 : C = 50
20 IF (A < B) AND (B = C) THEN 40
30 PRINT "PRIMA ESPRESSIONE FALSA" : GO TO 50
40 PRINT "PRIMA ESPRESSIONE VERA"
50 A$ = "X" : B$ = "Y" : C$ = "Z"
60 IF (A$ < > B$) AND (B$ < > C$) THEN 80
70 PRINT "SECONDA ESPRESSIONE FALSA" : GO TO 90
80 PRINT "SECONDA ESPRESSIONE VERA"
90 A = 42 AND 38
```

Risultato:

Sulla base dei valori impostati alla linea 10, l'espressione logica della linea 20 risulterà falsa, per cui verrà stampato il messaggio alla linea 30:
PRIMA ESPRESSIONE FALSA
Sulla base dei valori stringa impostati alla linea 50, l'espressione logica alla linea 60 risulterà vera, per cui verrà stampato il messaggio alla linea 80:
SECONDA ESPRESSIONE VERA
Il valore che verrà memorizzato in A (linea 90) sarà uguale a 34.
Infatti: 42_D equivale a 101010_B e 38_D equivale a 100110_B. L'AND tra i due dà come risultato 100010_B, che vale appunto 34_D.

Note:
— La tavola della verità dell'operatore AND è la seguente:
(V = vero = 1; F = Falso = 0).
— Su TI 99/4A e MZ-700 al posto della parola AND si usa l'operatore algebrico "*", che assume lo stesso significato.

X	Y	X AND Y
V	V	V
V	F	F
F	V	F
F	F	F

	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

OR (Operatore Logico)

Viene usato nelle espressioni logiche per legare due condizioni. L'espressione sarà vera se almeno una condizine è vera, altrimenti l'espressione risulterà falsa.

Esempio:

```
10 A = 1 : B = 30 : C = 50
20 IF (A < B) OR (B = C) THEN 40
30 PRINT "PRIMA ESPRESSIONE FALSA": GO TO 50
```

```
40 PRINT "PRIMA ESPRESSIONE VERA"
50 A$ = "X" : B$ = "Y" : C$ = "Z"
60 IF (A$ < > B$) OR (B$ = C$) THEN 80
70 PRINT "SECONDA ESPRESSIONE FALSA" : GO TO 90
80 PRINT "SECONDA ESPRESSIONE VERA"
90 A = 42 OR 38
```

Risultato:

Sulla base dei valori impostati alla linea 10, l'espressione logica della linea 20 risulterà vera, per cui verrà stampato il messaggio alla linea 40:
PRIMA ESPRESSIONE VERA
Sulla base dei valori stringa impostati alla linea 50, l'espressione logica alla linea 60 risulterà vera, per cui verrà stampato il messaggio alla linea 80:
SECONDA ESPRESSIONE VERA
Il valore che verrà memorizzato in A alla linea 90 sarà uguale a 46.
Infatti 42_D equivale a 101010_B e 38_D equivale a 100110_B. L'OR tra i due dà come risultato 101110_B, che equivale appunto a 46_D.

Note:
— La tavola della verità dell'operatore OR è la seguente: (V = Vero = 1; F = Falso = 0).
— Su TI 99/4A e MZ-700 al posto della parola OR si usa l'operatore algebrico "+", che assume lo stesso significato.

X	Y	X OR Y
V	V	V
V	F	V
F	V	V
F	F	F

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

XOR (Operatore Logico)

Viene usato nelle espressioni logiche per legare due condizioni. L'espressione sarà falsa se entrambe le condizioni sono vere oppure se entrambe le condizioni sono false. Negli altri casi l'espressione risulterà falsa.

Esempio:

```
10 A = 4 : B = 6
20 IF A = 3 XOR B = 6 THEN 40
30 PRINT "ESPRESSIONE FALSA" : END
40 IF A = 4 XOR B = 6 THEN 60
50 PRINT "QUESTO VERIFICA XOR" : END
60 PRINT "ESPRESSIONE VERA" : END
```

Risultato:

Sulla base dei valori impostati alla linea 10, l'espressione logica della linea 20 risulterà vera. La successiva esecuzione della linea 40 produrrà la stampa del messaggio della linea 50:
QUESTO VERIFICA XOR
perchè l'espressione logica alla linea 40 risulta falsa (entrambe le condizioni sono vere).

Note:
— La tavola della verità dell'operatore XOR è la seguente: (V = Vero = 1; F = Falso = 0).
— IF A = 3 XOR B = 6 può essere simulato da: IF (A = 3 OR B = 6) AND NOT (A = 3 AND B = 6).

X	Y	X OR Y
V	V	F
V	F	V
F	V	V
F	F	F

IMP (Operatore Logico)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Viene usato nelle espressioni logiche per legare due condizioni. L'espressione sarà falsa quando la prima condizione è vera e la seconda è falsa. Negli altri casi l'espressione risulterà vera.

Esempio:

```
10 A = 4 : B = 6
20 IF A = 3 IMP B = 6 THEN 40
30 PRINT "ESPRESSIONE FALSA" : END
40 IF A= 4 IMP B = 5 THEN 60
50 PRINT "QUESTO VERIFICA IMP" : END
60 PRINT "ESPRESSIONE VERA" : END
```

Risultato:

Sulla base dei valori impostati alla linea 10, l'espressione logica della linea 20 risulterà vera. La successiva esecuzione della linea 40 produrrà la stampa del messaggio alla linea 50:
QUESTO VERIFICA IMP
perchè l'espressione logica alla linea 40 risulta falsa (prima condizione vera e seconda falsa).

Note:
— La tavola della verità dell'operatore IMP è la seguente: (V = Vero = 1; F = Falso = 0).

X	Y	X IMP Y
V	V	V
V	F	F
F	V	V
F	F	V

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

EQV (Operatore Logico)

Viene usato nelle espresioni logiche per legare due condizioni. L'espressione sarà vera se entrambe le condizioni sono vere oppure se entrambe le condizioni sono false. Negli altri casi l'espressione risulterà falsa.

Esempio:

```
10 A = 4 : B = 6
20 IF A = 3 EQV B = 5 THEN 40
30 PRINT "ESPRESSIONE FALSA" : END
40 IF A= 4 EQV B = 7 THEN 60
50 PRINT "QUESTO VERIFICA EQV" : END
60 PRINT "ESPRESSIONE VERA" : END
```

Risultato:

Sulla base dei valori impostati alla linea 10, l'espressione logica della linea 20 risulterà vera. La successiva esecuzione della linea 40 produrrà la stampa del messaggio alla linea 50:
QUESTO VERIFICA EQV
perchè l'espressione logica alla linea 40 risulta falsa (prima condizione vera e seconda falsa).

Note:
— La tavola della verità dell'operatore EQV è la seguente: (V = Vero = 1; F = Falso = 0).

X	Y	X EQV Y
V	V	V
V	F	F
F	V	F
F	F	V

MATRICI

Una "matrice" (o schiera) è un insieme di variabili dello stesso tipo, aventi tutte lo stesso nome ma distinguibili tramite il valore di uno o più indici indicati (tra parentesi) dopo il nome. Ad esempio, se A è il nome di una matrice ad una dimensione, A (0) è il suo primo elemento, A (1) è il secondo, ecc. (supponendo che il valore minimo degli indici sia 0).

La sintassi del BASIC non pone alcuna limitazione al numero di dimensioni che si possono usare per le matrici, ma in pratica il numero di dimensioni è limitato dalla quantità di spazio di memoria libero disponibile per la memorizzazione delle variabili di matrice.

Per definire una matrice è necessario:

Attribuire un nome alla matrice (le stesse regole specificate per i nomi delle variabili semplici sono anche valide per i nomi delle matrici).

Stabilire il valore massimo e il valore minimo degli indici.

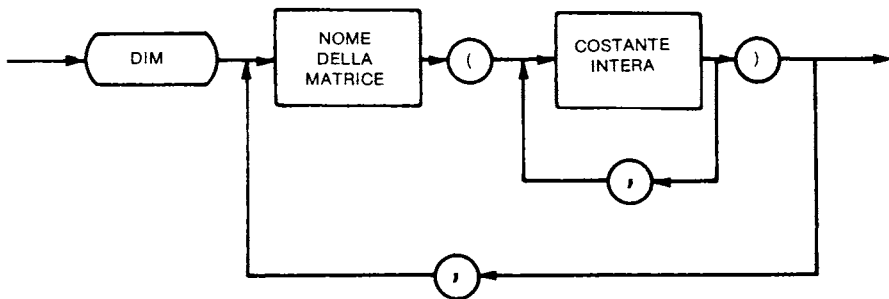
Per fare ciò è necessario scrivere una istruzione DIM ed eventualmente una istruzione OPTION BASE.

Possono anche essere definite matrici alfanumeriche, il cui nome dovrà avere il suffisso "$".

DIM

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Specifica il nome di una matrice, il numero delle sue dimensioni e il valore massimo dell'indice per ogni dimensione, e alloca spazio in memoria per la matrice stessa.

Formalismo sintattico:

Esempio:

```
10 DIM A (3, 2)
20 FOR I = 0 TO 3
30 FOR J = 0 TO 2
40 READ A (I, J)
50 NEXT J
60 NEXT I
70 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
```

Risultato:

Alla linea 10 viene specificata una matrice bidimensionale di 4 x 3 = 12 elementi. Le linee 20 e 30 iniziano i "loop", rispettivamente, di riga e di colonna. La linea 40 legge i valori dell'istruzione data e li assegna agli elementi della matrice. Le linee 50 e 60 chiudono i "loop" rispettivamente verso la linea 30 e la linea 20. Alla fine dell'esecuzione la matrice conterrà:

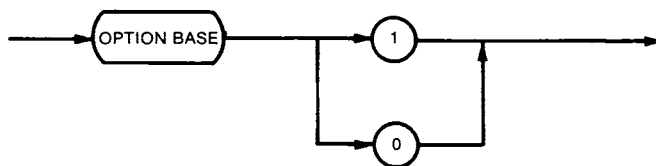
(0,0) 1	(0,1) 2	(0,2) 3
(1,0) 4	(1,1) 5	(1,2) 6
(2,0) 7	(2,1) 8	(2,2) 9
(3,0) 10	(3,1) 11	(3,2) 12

	NUMERO DIMENSIONI MAX	LUNGHEZZA DEL NOME DELLA MATRICE
TI 99/4A	3	1 car.
VIC 20	?	?
CBM 64	255	come variabili
ZX81	?	1 car.
ZX spectrum	255	1 car.
MZ 700	4	2 car.
IBM P.C.	qualsiasi	come variabili
Olivetti M20	qualsiasi	come variabili
Apple	88	2 car.

OPTION BASE

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Dichiara il valore minimo degli indici delle matrici.

Formalismo sintattico:

Esempio:

```
10 DIM A (3, 2)
20 OPTION BASE 1
30 FOR I = 1 TO 3
40 FOR J = 1 TO 2
50 READ A (I, J)
60 NEXT J
70 NEXT I
80 DATA 1, 2, 3, 4, 5, 6
```

Risultato:

Alla linea 10 viene specificata una matrice bidimensionale di 4 x 3 = 12 elementi. La linea 20 limita a 1 il valore minimo degli indici, riducendo la matrice a 3 x 2 = 6 elementi. Le linee 30 e 40 iniziano i "loop", rispettivamente, di riga e di colonna. La linea 50 leggi i valori della istruzione DATA e li assegna agli elementi della matrice. Le linee 60 e 70 chiudono i "loop" rispettivamente con la linea 40 e la linea 30. Alla fine dell'esecuzione la matrice conterrà:

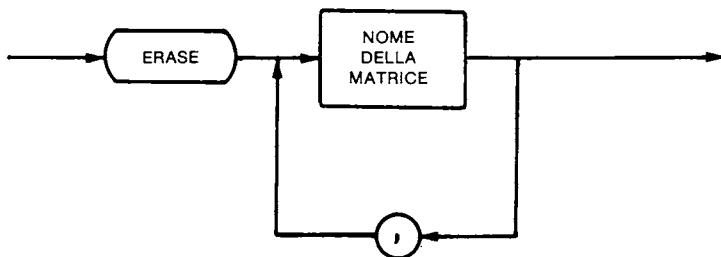
(1.1) 1	(1,2) 2
(2,1) 3	(2,2) 4
(3,1) 5	(3,2) 6

Note:
— Una istruzione OPTION BASE agisce su *tutte* le matrici definite nell'ambito del programma. Non è cioè possibile, in un programma, definire alcune matrici con valore minimo zero degli indici e altre matrici con valore minimo 1.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

ERASE

Libera lo spazio riservato a una o più matrici e rende disponibili i relativi nomi per nominare altre variabili del programma.

Formalismo sintattico:

Esempio:

```
10 PRINT FRE (" ")
20 DIM A (200, 50)
30 PRINT FRE (" ")
40 ERASE A
50 DIM B (20, 5)
60 PRINT FRE (" ")
```

Risultato:

LINEA 10: Stampa la quantità di memoria ancora a disposizione. Se venisse stampato:
63100
vuol dire che in memoria vi sono ancora 63100 byte a disposizione.

LINEA 20: Alloca spazio in memoria per la matrice A di 201 x 51 = 10251 elementi.

LINEA 30: Stampa la quantità di memoria ancora a disposizione. Nel nostro caso verrebbe stampato:
22096.

LINEA 40: Cancella la matrice A e libera lo spazio di memoria allocato alla matrice stessa.

LINEA 50: Alloca spazio in memoria per la matrice B di 21 x 6 = 126 elementi.

LINEA 60: Stampa la quantità di memoria ancora a disposizione. Nel nostro caso verrebbe stampato: 62596.

CAPITOLO 2

COMANDI E ISTRUZIONI PIU' COMUNI

a) Comandi

NEW
LIST
RUN
AUTO
NUMBER
RENUM
RESEQUENCE
TRON
TRACE
TROFF
NOTRACE
UNTRACE
BREAK
UNBREAK
CONTINUE
SAVE
BSAVE
STORE
VERIFY
LOAD
OLD
BLOAD
RECALL

b) Istruzioni per l'assegnamento di valori a variabili

LET
CLR
CLEAR

c) Istruzioni di Input/Output

PRINT
WRITE
INPUT
GET
CALL KEY
INKEY$
LINE INPUT
PRINT USING
DATA
READ
RESTORE

d) Istruzioni condizionali

IF ... THEN
IF ... GOTO
IF ... GOSUB

e) Istruzioni che variano la sequenza di esecuzione

GOTO
STOP
END
GOSUB
RETURN
POP
ON ... GOTO
ON ... GOSUB
FOR
NEXT

f) Varie

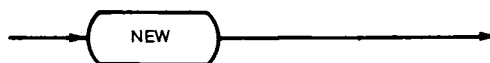
REM
RANDOMIZE
RND
POKE

g) Modo virgolette

NEW (Comando)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Cancella il programma residente in memoria e le variabili, permettendo all'utente di introdurre un nuovo programma. Azzera il bit di "trace" e chiude tutti i file di dati.

Formalismo sintattico:

Esempio:

```
  10 REM PROGRAMMA CHE DOPO L'ESECUZIONE
  20 REM VIENE CANCELLATO DALLA MEMORIA
  30 A = 1
  40 B = 5
  50 C = (A * B)/2
  60
   :
1000 NEW
```

Risultato:

Il programma viene eseguito fino alla linea 1000. A questo punto viene cancellato dalla memoria. Si può verificare ciò con un comando LIST.

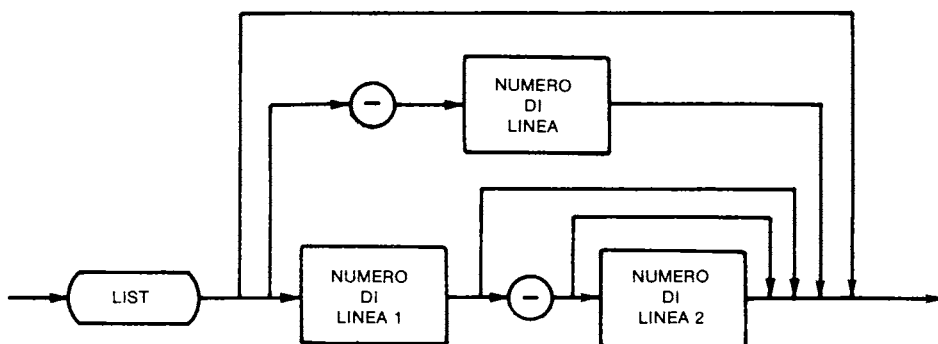
Note:

— Può essere usato sia in modo immediato che in modo differito.
— Quando si carica un programma da una unità periferica, se il caricamento va a buon fine il programma precedentemente residente in memoria viene cancellato per lasciar posto al nuovo, per cui non serve dare il comando NEW.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

LIST (Comando)

Lista sul video le linee di programma attualmente in memoria.

Formalismo sintattico:

Esempi:

In modo immediato:

1. LIST 10
   Lista la linea 10

2. LIST 10-50
   Lista tutte le linee numerate da 10 a 50 incluse.

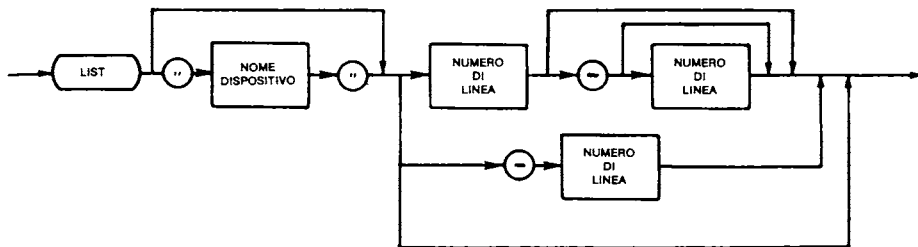
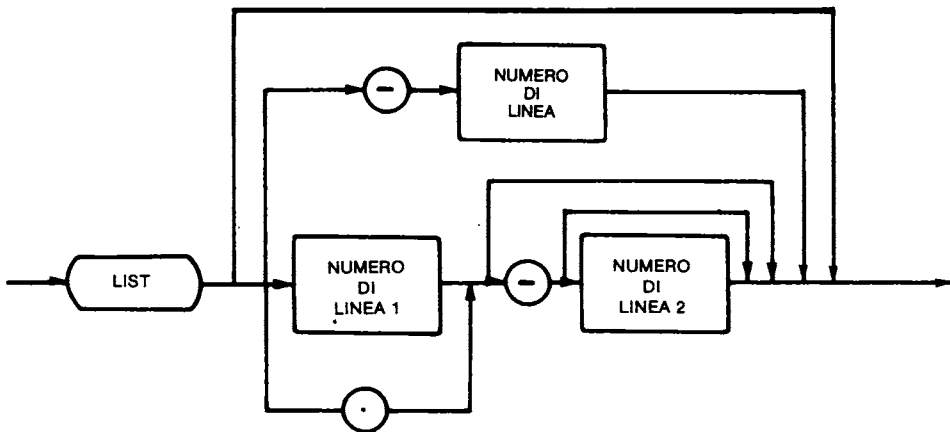
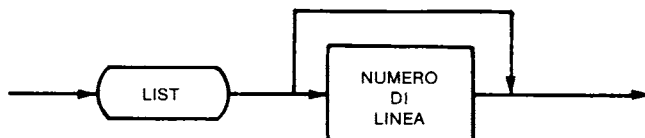
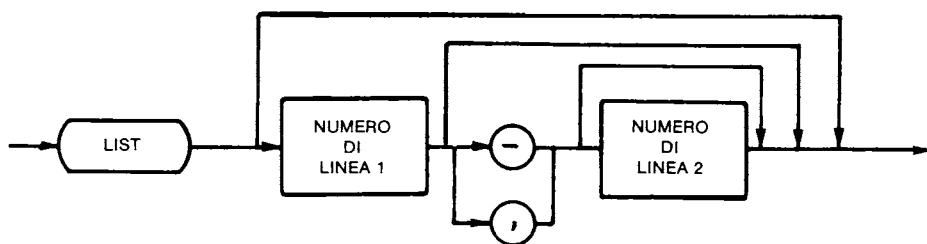
3. LIST-50
   Lista tutte le linee dall'inizio del programma fino alla 50 inclusa.

4. LIST 50-
   Lista tutte le linee dalla 50 alla fine del programma.

5. LIST
   Lista tutto il programma

In modo differito:

6. 10 INPUT X:IF X = 1 THEN LIST 10-100
   Lista tutte le linee dalla 10 alla 100 incluse se il valore digitato in risposta all'istruzione INPUT alla linea 10 è uguale a 1.

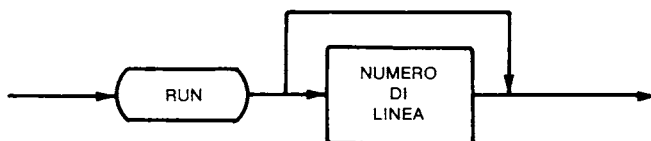
Variazione per: TI 99/4A

Variazione per: IBM P.C. - M20

Variazione per: ZX 81 - ZX Spectrum

Variazione per: Apple II

Note:

- In dipendenza del sistema, il listato può essere temporaneamente fermato in modo immediato, premendo il tasto BREAK, oppure il tasto STOP, oppure il tasto CONTROL seguito dalla lettera C, oppure il tasto CONTROL seguito dalla lettera S, ecc.
- Il numero di linea è una costante numerica che può assumere i valori da 0 a 65.535. Il numero di linea 1 deve essere inferiore o uguale al numero di linea 2.

RUN (Comando)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Lancia l'esecuzione del programma residente in memoria. In certi casi può anche eseguire un programma residente su supporto esterno, trasferendolo dapprima in memoria.

Formalismo sintattico:

Esempio:

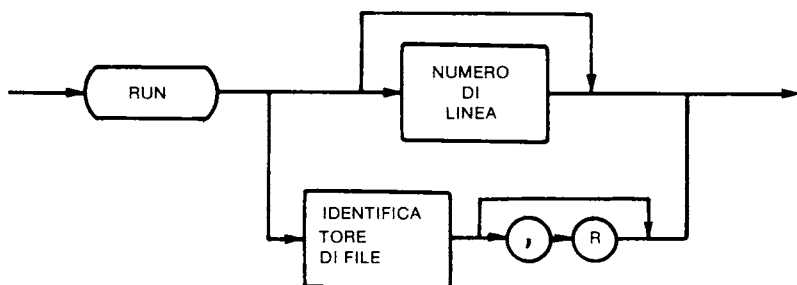
```
  10 INPUT Q$
  20 IF Q$ = "SI" THEN RUN : END
  30 RUN 1000
  40 END
     :
1000 REM * SOTTOPROGRAMMA *
```

Risultati:

LINEA 10: attende che venga digitato qualcosa in ingresso e lo assegna alla variabile stringa Q$.

LINEA 20: se la stringa impostata alla linea 10 contiene il valore SI, il programma corrente viene eseguito dal primo numero di linea (il più basso).

LINEA 30: esegue il programma corrente dal numero di linea 1000 in avanti.

Variazioni per: IBM P.C. - M20

Esempio:
RUN "CAS 1: PG1", R

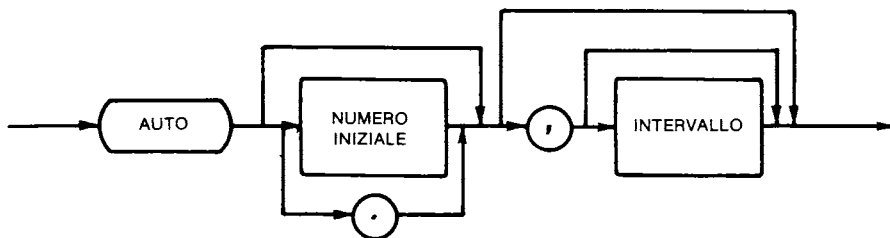
Risultato:
Carica il programma PG1 da cassetta e lo esegue, mantenendo aperti gli eventuali file.

Note:
— RUN può essere usato sia in modo immediato che in modo differito.
— Se il numero di linea specificato dopo RUN non dovesse esistere nel programma, verrà emesso un opportuno messaggio di errore.
— RUN inizializza a zero tutte le variabili numeriche e a spazi tutte le variabili stringa, azzera tutti i puntatori e gli stacks, e chiude tutti i files.
Vedere anche Appendice A (per IDENTIFICATORE DI FILE).

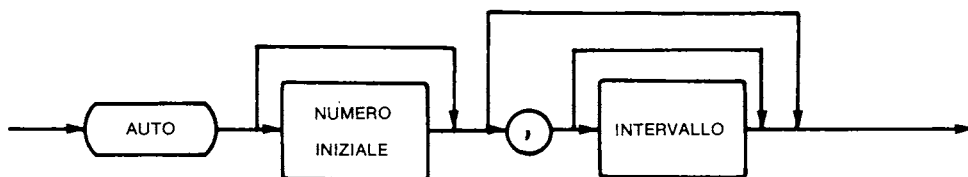
	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

AUTO (Comando)

Fa iniziare la numerazione automatica delle linee.

Formalismo sintattico:

Esempi:

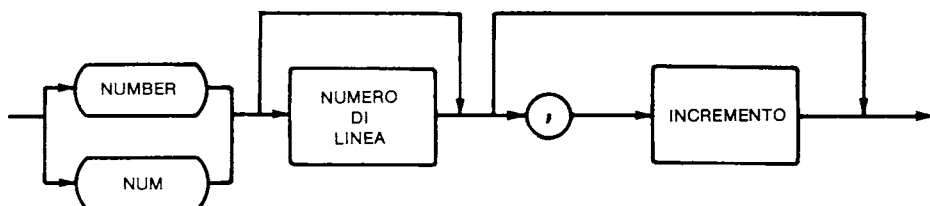
1) AUTO 10,10
2) AUTO 20,5
Nell'esempio 1 il comando AUTO, partendo dalla linea 10, genera automaticamente i numeri di linea successivi con intervalli di 10 (10, 20, 30, 40, ...ecc.).
Nell'esempio 2 il comando AUTO, partendo dalla linea 20, genera automaticamente i numeri di linea successivi con intervalli di 5 (20, 25, 30, 35, ...ecc.).

Variazione per MZ 700:

Note:
— I valori di "Default" per NUMERO INIZIALE e INTERVALLO sono rispettivamente 10 e 10.
— NUMERO INIZIALE e INTERVALLO sono costanti numeriche che possono assumere valori compresi tra zero e 65.536.
— In dipendenza del sistema, la numerazione automatica può essere interrotta premendo il tasto BREAK, oppure il tasto STOP, oppure il tasto CONTROL seguito dal tasto C, ecc.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

NUMBER (Comando)

Fa iniziare la numerazione automatica delle linee.

Formalismo sintattico:

Esempi:

1) NUMBER 10
2) NUMBER ,5
3) NUMBER 50,50

Risultati:

1) Genera i numeri di linea in modo automatico partendo da 10 con incrementi di 10 (10, 20, 30, 40, ... ecc.).
2) Genera i numeri di linea in modo automatico partendo da 100 con incrementi di 5 (100, 105, 110, 115, ... ecc.).
3) Genera i numeri di linea in modo automatico partendo da 50 con incrementi di 50 (50, 100, 150, 200, ...ecc.).

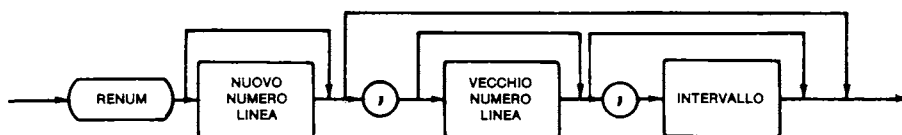
Note:

— I valori di "default" per NUMERO DI LINEA e INCREMENTO sono rispettivamente di 100 e 10.
— NUMERO DI LINEA e INCREMENTO sono costanti numeriche che possono assumere valori compresi fra zero e 32767.
— La numerazione automatica delle linee può essere interrotta premendo il tasto ENTER subito dopo che è stato generato un numero di linea.

RENUM (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Modifica i numeri di linea del programma residente in memoria.

Formalismo sintattico:

Esempi:

1) RENUM
2) RENUM 300, 210, 15
3) RENUM 200,,5

Risultati:
Nell'esempio 1 le linee del programma che si trova in memoria vengono numerate partendo da 10 con intervallo 10:

10 20 30 40 ecc.

Nell'esempio 2 vengono numerate tutte le linee, con intervallo 15, partendo dalla linea 210 del programma in memoria che diventa la linea 300 della nuova numerazione:

300 315 330 345 ecc.

Nell'esempio 3 vengono numerate tutte le linee con intervallo 5 partendo con il numero di linea 200.

Note:

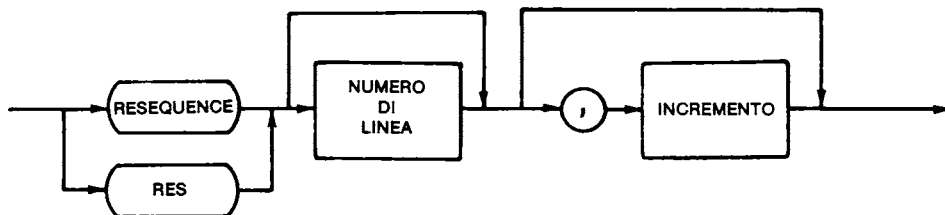
— Le istruzioni GOTO, GOSUB, ON...GOTO e ON...GOSUB vengono aggiornate opportunamente con i nuovi numeri di linea; in questo modo i salti nel programma rimangono validi.

— NUOVO NUMERO LINEA, VECCHIO NUMERO LINEA e INTERVALLO sono costanti numeriche con valori che possono essere compresi tra zero e 65.535.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

RESEQUENCE (Comando)

Modifica i numeri di linea del programma residente in memoria.

Formalismo sintattico:

Esempi:

1) RESEQUENCE
2) RES 300, 15
3) RES 200, 5

Risultati:

1) Tutte le linee del programma vengono rinumerate partendo da 10 con incremento di 10:
   (10, 20, 30, 40 ..., ecc.).

2) Tutte le linee del programma vengono rinumerate iniziando da 300 e con incrementi di 15:
   (300, 315, 330, 345, ..., ecc.).

3) Tutte le linee del programma vengono rinumerate iniziando da 200 con intervalli di 5:
   (200, 205, 210, 215, ..., ecc.).

Note:
— Le istruzioni GOTO, GOSUB, ON...GOTO e ON...GOSUB vengono aggiornate opportunamente con i nuovi numeri di linea; in questo modo i salti nel programma rimangono validi.
— NUMERO DI LINEA e INCREMENTO sono costanti numeriche con valori che possono essere compresi tra zero e 65.535.

TRON (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Produce una lista dei numeri di linea di tutte le istruzioni eseguite.

Formalismo sintattico:

Esempio:

```
 10 TRON
 20 PRINT "ANAGRAFICA"
 30 PRINT "CLIENTI"
 40 GO TO 100
    :
100 PRINT "INDIRIZZO"
    :
160 TROFF
170 END
```

Risultato: dopo un RUN verrà stampato:

<20> ANAGRAFICA
<30> CLIENTI
<40> <100> INDIRIZZO

Note:

— Il comando TRON può essere usato sia in modo immediato che come istruzione di programma.
— La lista prodotta può essere interrotta con l'utilizzo del comando TROFF.
— Il comando TRON è presente nella versione 1Z-013B del BASIC dello SHARP MZ-700, anche se il manuale in dotazione non ne fa menzione.

TRACE (Comando)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Produce una lista dei numeri di linea di tutte le istruzioni eseguite.

Formalismo sintattico:

Esempio:

```
 10 TRACE
 20 PRINT "ANAGRAFICA"
 30 PRINT "CLIENTI"
 40 GO TO 100
    :
100 PRINT "INDIRIZZO"
    :
160 NO TRACE
170 END
```

Risultato:

Dopo un RUN verrà stampato:
20 ANAGRAFICA
30 CLIENTI
#40 # 100 INDIRIZZO

Note:
— Il comando TRACE può essere usato sia in modo immediato che come istruzione di programma.
— La lista prodotta può essere interrotta con l'utilizzo del comando UNTRACE (per TI99/4A) o NOTRACE (per APPLE).

TROFF (Comando)

Arresta la lista iniziata con l'impostazione del comando TRON.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Formalismo sintattico:

Esempio:

```
 10 TRON
 20 PRINT "ANAGRAFICA"
 30 PRINT "CLIENTI"
 40 GO TO 100
     :
100 PRINT "INDIRIZZO"
     :
160 TROFF
170 PRINT "FINE"
180 END
```

Risultato:

dopo un RUN verrà stampato:
<20> ANAGRAFICA
<30> CLIENTI
<40> <100> INDIRIZZO
FINE

Note:
— Il comando TROFF può essere usato sia in modo immediato che come istruzione di programma.
— Il comando TROFF è presente nella versione 1Z-013B del BASIC dello SHARP MZ-700, anche se il manuale in dotazione non ne fa menzione.

NOTRACE (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Arresta la lista iniziata con l'impostazione del comando TRACE.

Formalismo sintattico:

Esempio:

```
 10 TRACE
 20 PRINT "ANAGRAFICA"
 30 PRINT "CLIENTI"
 40 GO TO 100
    :
100 PRINT "INDIRIZZO"
    :
160 NO TRACE
170 PRINT "FINE"
180 END
```

Risultato:

dopo un RUN verrà stampato:
20 ANAGRAFICA
30 CLIENTI
40 # 100 INDIRIZZO
FINE

Note:

— Il comando NOTRACE può essere usato sia in modo immediato che come istruzione di un programma.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

UNTRACE (Comando)

Arresta la lista iniziata con l'impostazione del comando TRACE.

Formalismo sintattico:

Esempio:

```
 10 TRACE
 20 PRINT "ANAGRAFICA"
 30 PRINT "CLIENTI"
 40 GO TO 100
     :
100 PRINT "INDIRIZZO"
     :
160 UNTRACE
170 PRINT "FINE"
180 END
```

Risultato:

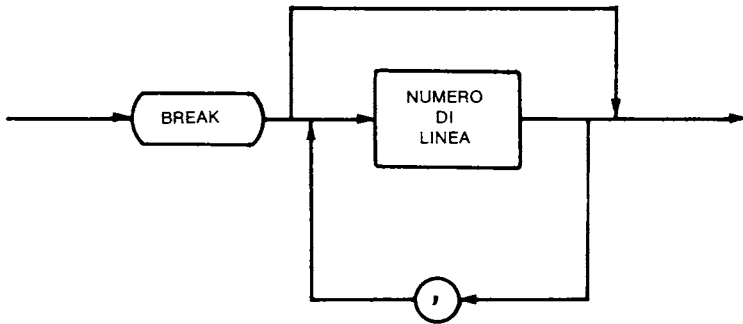
dopo un RUN verrà stampato:
20 ANAGRAFICA
30 CLIENTI
40 # 100 INDIRIZZO
FINE

Note:
— Il comando UNTRACE può essere usato sia in modo immediato che come istruzione di un programma.

BREAK (Comando)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Con tale comando si inseriscono nel programma, alle linee specificate, dei punti di arresto che possono servire per trovare eventuali errori.

Formalismo sintattico:

Esempi:

a) In modo differito:

```
100 A = 50
110 BREAK 120, 150
120 B = 15
130 PRINT A
140 PRINT B
150 PRINT A – B
```

Risultato:

Se si esegue questo programma, si otterrà la seguente stampa: *BREAKPOINT AT 120 e il programma si arresta. Si può farlo ripartire usando il comando CONTINUE.

b) **In modo immediato:**
digitando BREAK 110 quando si ha in memoria, ad esempio, il programma precedente, e dando quindi il RUN, si otterrà la stampa seguente:
* BREAKPOINT AT 110

Note:

— Negli altri elaboratori non esiste come comando, e al suo posto si usano dei tasti opportuni.

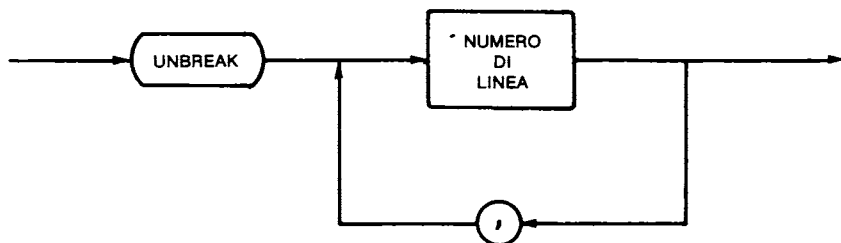
Per: TEXAS TI 99/4A	Tasto CLEAR
Per: VIC 20 CBM 64	Tasto RUN/STOP
Per: ZX 81 ZX Spectrum	Tasto BREAK
Per: MZ 700	Tasto SHIFT + Tasto BREAK
Per: IBM P.C.	Tasto CTRL + Tasto BREAK
Per: M20	Tasto CTRL + Tasto C

— Usato come istruzione, in modo differito, è possibile digitare solo la parola BREAK senza numero di linea. In tal caso l'istruzione stessa agisce come punto di arresto.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

UNBREAK (Comando-Istruzione)

Serve per eliminare i punti di arresto dalle linee specificate. Se non vengono specificati i numeri di linea, vengono eliminati tutti i punti di arresto definiti con una o più istruzioni (o comandi) BREAK.

Formalismo sintattico:

Esempi:

a) In modo differito:

```
100 A = 50
110 UN BREAK 120
120 B = 15
130 PRINT A
140 PRINT B
150 PRINT A - B
```

Risultato:

Se ora si digita: BREAK 120, 150 e poi RUN, si avrà che il programma non si fermerà alla linea 120, ma alla 150, per cui eseguirà le stampe seguenti:

```
50
15
* BREAKPOINT AT 150
```

b) In modo immediato:

```
100 A = 50
110 BREAK 120, 150
120 B = 15
130 PRINT A
140 PRINT B
150 PRINT A-B
```

Risultato:

Se si digita: UNBREAK 120 e poi RUN, il programma si comporterà esattamente come quello visto all'esempio a) precedente.

CONTINUE (o CONT) (Comando)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Serve per far ripartire un programma fermo su un punto di arresto definito con un comando (o una istruzione) BREAK oppure fermo per un opportuno utilizzo degli appositi tasti.

Formalismo sintattico:

Esempio:

```
100 PRINT "BATTI CONT PER CONTINUARE"
110 STOP
120 PRINT "IL PROGRAMMA PROSEGUE"
```

Risultato:

LA LINEA 100 fornisce un messaggio.
LA LINEA 110 interrompe temporaneamente l'esecuzione del programma e viene visualizzato il messaggio: BREAK in 110.
Battendo CONT e RETURN l'esecuzione del programma continua con l'istruzione successiva allo STOP, per cui viene stampata la frase: IL PROGRAMMA PROSEGUE.

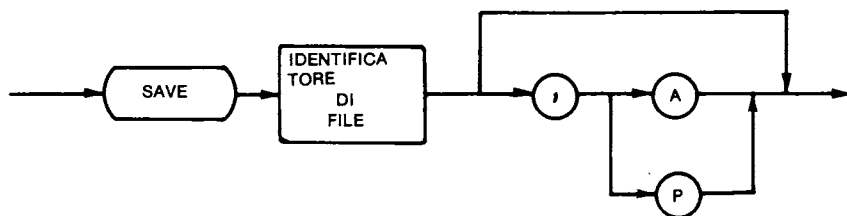
Variazione per: TI 99/4A
ZX Spectrum.

Note:
— Solitamente si usa CONT con STOP o con BREAK per la ricerca degli errori.
— CONT non è più valido se il programma viene modificato durante l'interruzione.

SAVE (Comando)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Serve per registrare su dispositivo esterno (disco o nastro) dei programmi BASIC.

Formalismo sintattico:

Esempio:

```
200 PRINT "PER SALVARE IL FILE BATTI – Y –"
210 INPUT A$
220 IF A$ = "Y" THEN SAVE "CARSOC"
230 END
```

Risultato:

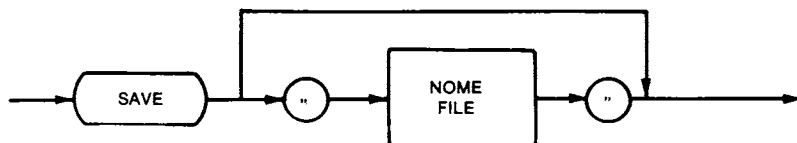
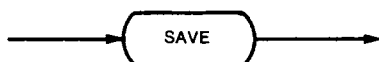
Le linee 200 e 210 consentono di assegnare un valore appropriato alla variabile A$. La linea 220, se il valore di A$ è "y", salva il programma su supporto esterno col nome "CARSOC".

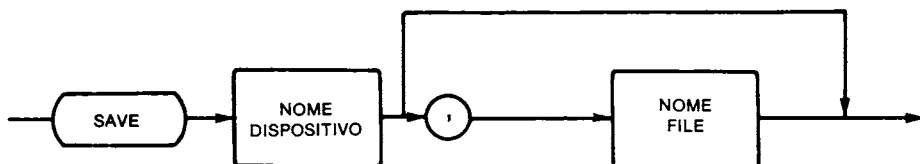
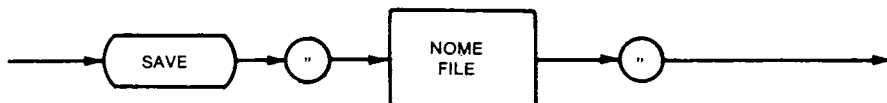
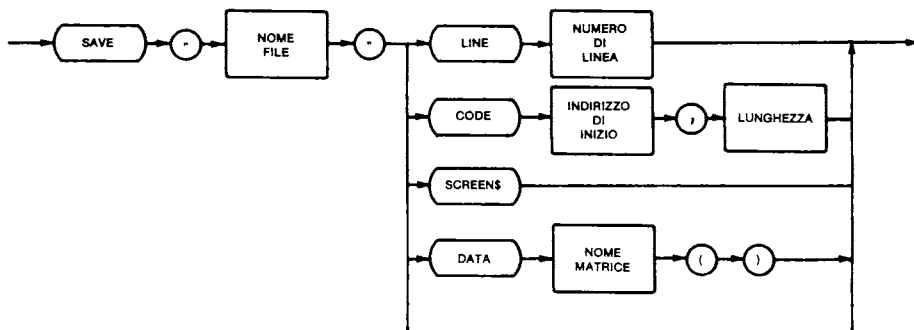
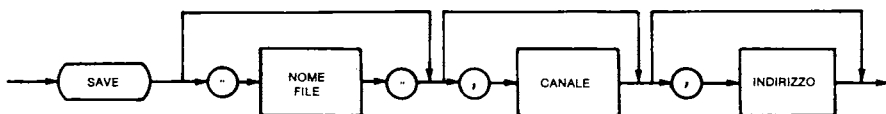
Note:
- — Un programma salvato con SAVE è normalmente caricato in memoria con il comando LOAD.
- — Il comando SAVE può essere utilizzato sia in modo immediato che come istruzione di programma.
- — Con l'opzione A, il programma viene salvato in formato ASCII, mentre normal-

mente viene salvato in formato binario compresso.

— Con l'opzione P il programma viene salvato in formato binario codificato, così che non può più venir listato o modificato. È un modo per proteggere il proprio software.

Vedere anche Appendice A (per IDENTIFICATORE DI FILE e NOME DISPOSITIVO).

Variazione per : MZ 700

Variazione per: APPLE

Variazione per: TI 99/4A

Variazione per: ZX 81

Variazione per ZX Spectrum

Variazione per: VIC 20 e CBM 64

Note:

— Il nome del file è un'espressione stringa con un numero massimo di 8 caratteri.

— L'opzione LINE per ZX Spectrum permette di far conoscere al sistema il numero di linea da cui far partire automaticamente il programma quando verrà caricato con un'istruzione LOAD.

— L'opzione CODE per ZX Spectrum consente di salvare blocchi di memoria, ed è perciò utilizzato per salvare programmi in linguaggio macchina. INDIRIZZO DI INIZIO e LUNGHEZZA sono espressioni numeriche che possono aver valori da 0 a 65.535 (come anche NUMERO DI LINEA dell'opzione LINE).

— L'opzione SCREEN$ per ZX Spectrum consente di salvare la configurazione attuale dell'immagine video.

— L'opzione DATA per ZX Spectrum consente di salvare il contenuto di matrici definite nel programma.

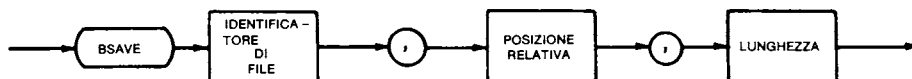
— CANALE (per CBM 64) è un'espressione numerica che può assumere i seguenti valori:
1 = nastro
8 = disco

— INDIRIZZO (per CBM 64) è un'espressione numerica che può assumere i seguenti valori:
1 = salva il programma a partire da una locazione diversa da quella standard.
2 = viene posto un segnale di fine programma dopo la registrazione del programma stesso.
3 = combina le funzioni viste per i valori 1 e 2.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

BSAVE (Comando)

Salva porzioni della memoria dell'elaboratore sull'unità specificata (per esempio un programma in linguaggio macchina).

Formalismo sintattico:

Esempio:

```
100 DEF SEG = &HB700
110 BSAVE "PIPPO", 0, &H250
```

Risultato:

Viene salvato un programma in linguaggio macchina,con nome PIPPO, dalla posizione specificata alla linea 100, con posizione relativa 0 per una lunghezza di 250 (esadecimale) byte.

Note:

— POSIZIONE RELATIVA è un'espressione numerica con valori compresi tra 0 e 65535.

— LUNGHEZZA è un'espressione numerica con valori compresi tra 1 e 65535.

— Il comando BSAVE può essere dato sia in modo immediato che in modo differito.

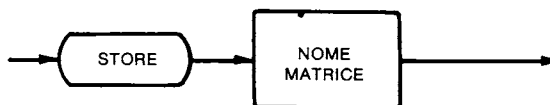
— Il comando BSAVE deve essere preceduto da una istruzione DEFSEG che definisce l'indirizzo di inizio del segmento di memoria interessato.

Vedere anche Appendice A (per IDENTIFICATORE DI FILE).

STORE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Salva una matrice su un supporto esterno (cassetta).

Formalismo sintattico:

Esempio:

```
 10 DIM MESE$ (12)
 20 MESE$ (1) = "GENNAIO"
 30 MESE$ (2) = "FEBBRAIO"
 40 MESE$ (3) = "MARZO"
    :
130 MESE$ (12) = "DICEMBRE"
200 STORE MESE$
```

Risultato:

LA LINEA 10 dimensiona la matrice MESE$ con 13 elementi (l'elemento MESE$ (0) non viene usato).
LE LINEE DALLA 20 ALLA 130 assegnano agli elementi della matrice i nomi dei mesi.
LA LINEA 200 memorizza la matrice MESE$ su cassetta.

Note:

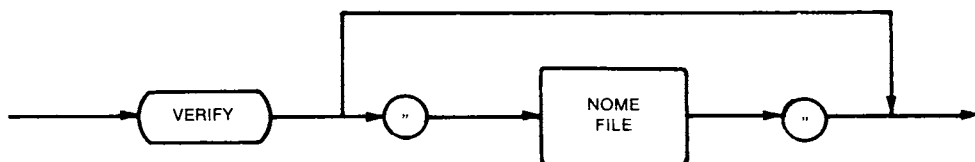
— L'istruzione RECALL consente di riportare la matrice dalla cassetta in memoria.

— Una matrice può essere memorizzata su cassetta (STORE) e riportata in memoria (RECALL) anche con nomi differenti, ma la dimensione della matrice deve essere la stessa.

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

VERIFY (Comando)

Viene usato per avere la conferma che il programma sia stato correttamente registrato su nastro.

Formalismo sintattico:

Esempio: a) In modo differito:

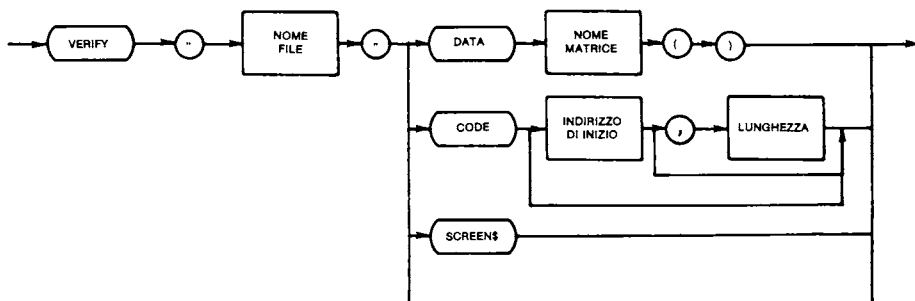
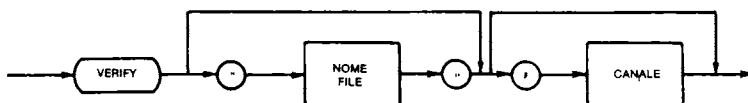
```
100 SAVE "PROG1"
110 VERIFY "PROG1"
```

Risultato:

LA LINEA 100 salva il file PROG1 su cassetta.
LA LINEA 110 controlla che il file sia stato memorizzato correttamente.

b) In modo immediato:
>SAVE "PROG1" ENTER
>VERIFY "PROG1" ENTER

Risultato:
è esattamente lo stesso dell'esempio a) precedente.

Variazione per: ZX Spectrum

Variazione per: VIC 20
CBM 64

Note:
— Se non è specificato il NOME FILE, verrà verificato il primo programma incontrato.

— Se si vuol verificare il programma appena salvato su nastro, si dovrà prima riavvolgere il nastro dall'inizio.

— Nel TI 99/4A non esiste il comando VERIFY isolato, ma la funzione di verifica è svolta nella complessa procedura di SAVE.

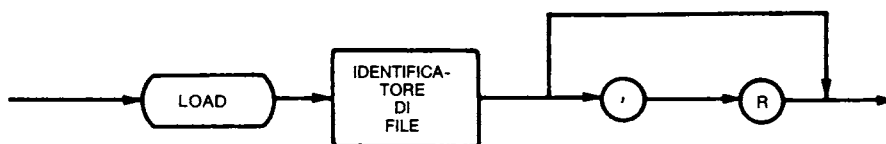
— Per quanto riguarda le opzioni DATA, CODE e SCREEN$ e i simboli INDIRIZZO DI INIZIO, LUNGH. e CANALE vedere le note al comando SAVE.

VEDERE ANCHE APPENDICE A.

	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

LOAD (Comando)

Trasferisce un programma da supporto esterno (disco o nastro) in memoria, ed eventualmente ne lancia l'esecuzione.

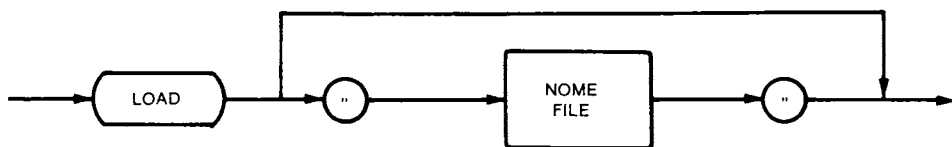
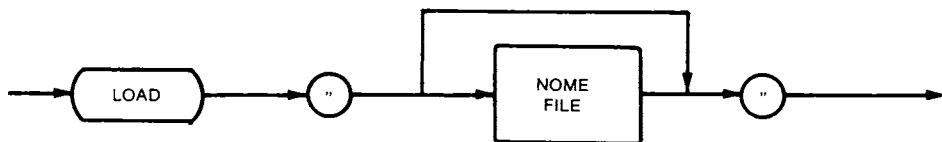
Formalismo sintattico:

Esempio:

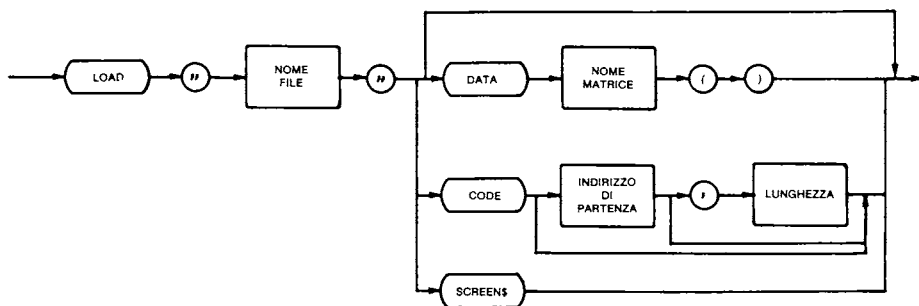
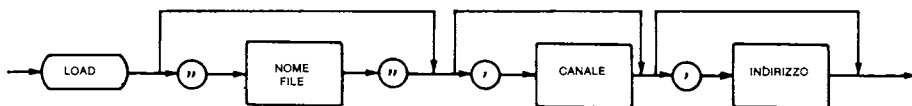
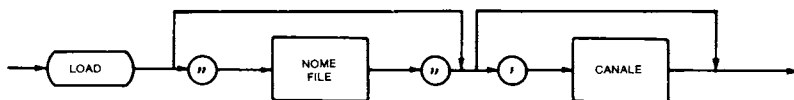
```
100 INPUT A$
110 IF A$ = "YES" THEN LOAD "CARSOC"
120 END
```

Risultato:

LA LINEA 100 assegna l'input alla variabile A$.
LA LINEA 110, se l'input era stato "YES" carica in memoria, da supporto esterno, il programma chiamato CARSOC.

Note:
— Il comando LOAD può essere usato sia in modo immediato che come istruzione di programma.
— Quando viene eseguito il comando LOAD, viene perso il precedente contenuto della memoria.
— L'opzione R fa in modo che il programma venga eseguito immediatamente dopo il suo caricamento.

Variazione per: MZ 700

Variazione per: ZX 81

Variazione per: Apple

Variazione per: ZX Spectrum

Variazione per: CBM 64

Variazione per: VIC 20

Note:

— Il nome del file è un'espressione stringa con un numero massimo di 8 caratteri.

— L'opzione CODE per ZX Spectrum consente di caricare blocchi di memoria o programmi in linguaggio macchina, salvati con la stessa opzione CODE del comando SAVE. INDIRIZZO DI PARTENZA e LUNGH. sono espressioni numeriche che possono assumere valori da 0 a 65.535.

— L'opzione SCREEN$ per ZX Spectrum consente di caricare in memoria l'immagine video che era stata preventivamente salvata con la stessa opzione SCREEN$ del comando SAVE.

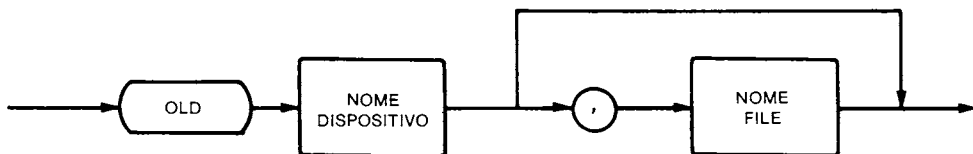
— L'opzione DATA per ZX Spectrum consente di caricare in memoria il contenuto di matrici: tale contenuto era stato preventivamente salvato con la stessa opzione DATA del comando SAVE.
— CANALE (per CBM64) è un'espressione numerica che può assumere i seguenti valori:
1 = nastro
8 = disco
— INDIRIZZO (per CBM 64) è un'espressione numerica che può assumere i seguenti valori:
1 = carica il programma nello stesso segmento di memoria da cui è stato salvato.

Vedere anche Appendice A (per IDENTIFICATORE DI FILE).

OLD (Comando)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Trasferisce un programma da supporto esterno (disco o cassetta) in memoria.

Formalismo sintattico:

Esempio:
OLD CS1

Risultato:

Se il programma da caricare viene letto in modo corretto, il sistema effettuerà le seguenti stampe:

* REWIND CASSETTE TAPE CS1
THEN PRESS ENTER
* PRESS CASSETTE PLAY CS1
THEN PRESS ENTER
* READING
* DATA OK
* PRESS CASSETTE STOP CS1
THEN PRESS ENTER

Se il programma è inesistente o non viene letto in modo corretto, al posto della frase: DATA OK, verrà stampato:

* ERROR-NO DATA FOUND
PRESS R TO READ
PRESS E TO EXIT

oppure

* I/O ERROR 56

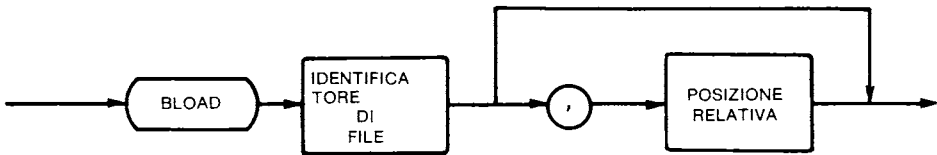
Note:

— Il comando OLD, come anche il comando SAVE e VERIFY, per il TI 99/4A, può essere usato solo in modo immediato.
— Quando viene eseguito correttamente il comando OLD, viene perso il precedente contenuto della memoria.
— Per NOME DISPOSITIVO vedere Appendice A.

BLOAD (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Carica in memoria un file ad immagine della memoria (per esempio un programma in linguaggio macchina).

Formalismo sintattico:

Esempio:

```
100 DEF SEG = &HB700
110 BLOAD "PIPPO", 0
```

Risultato:

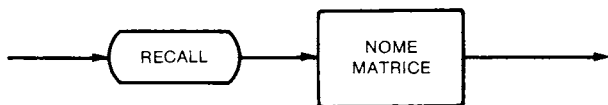
Viene caricato in memoria il file PIPPO a partire dall'indirizzo assoluto di memoria specificato alla linea 100, e con posizione relativa 0.

Note:
— Il comando BLOAD può essere usato sia in modo immediato che in modo differito.
— POSIZIONE RELATIVA è un'espressione numerica con valori compresi tra 0 e 65.535.
— Il comando BSAVE deve essere preceduto da una istruzione DEF SEG che specifichi l'indirizzo di partenza del segmento di memoria dove effettuare il caricamento.
Vedere anche Appendice A (per IDENTIFICATORE DI FILE).

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

RECALL (Istruzione)

Richiama una matrice da un supporto esterno (cassetta).

Formalismo sintattico.

Esempio:

```
200 DIM MESE$ (12)
210 RECALL MESE$
220 FOR I = 1 TO 12
230 PRINT I, MESE$ (I)
240 NEXT I
```

Risultato:

LA LINEA 200 dimensiona la matrice MESE$ con 13 elementi (l'elemento MESE$ (0) non viene utilizzato).
LA LINEA 210 richiama una matrice da supporto esterno e la assegna a MESE$.
Le linee successive effettueranno la stampa dei dodici mesi nel modo seguente:

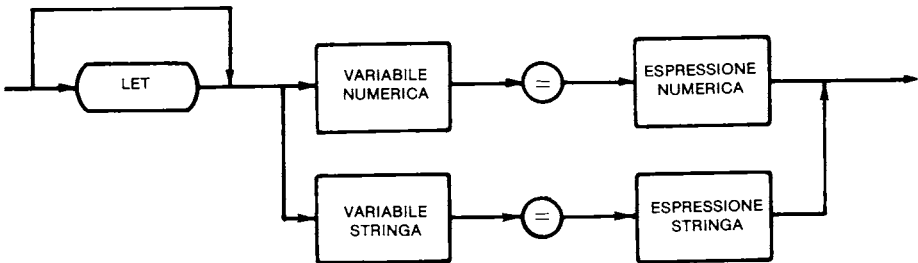
```
 1          GENNAIO
 2          FEBBRAIO



12          DICEMBRE
```

Note:
— Per le modalità di salvataggio della matrice vedere il comando STORE.
— Le matrici possono venir salvate e richiamate con nomi diversi, ma le dimensioni devono essere le stesse.

LET (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Assegna un valore ad una variabile.

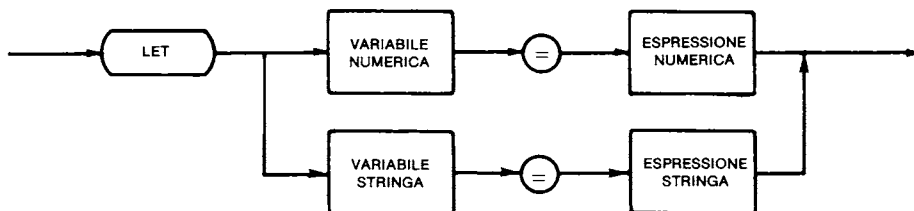
Formalismo sintattico:

Esempio:

```
10 LET A$ = "ABCD"
20 A = 2
30 LET B = A * 3
40 LET C = (A + B)/4
50 LET B$ = A$
60 PRINT A$, A, B, C, B$
70 END
```

Risultato:

LA LINEA 10 assegna la stringa ABCD alla variabile A$.
LA LINEA 20 assegna il valore 2 alla variabile A.
LE LINEE 30 e 40 assegnano il risultato dell'espressione matematica alle variabili B e C.
LA LINEA 50 assegna alla variabile B$ il contenuto della variabile A$.
LA LINEA 60 provoca la stampa:

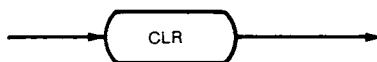
ABCD 2 6 2 ABCD.

Variazione per: ZX 81
ZX Spectrum

Note:
— Tranne che nei SINCLAIR (ZX 81 e ZX Spectrum) la parola LET è facoltativa, e può quindi essere omessa (vedere la linea 20 dell'esempio).

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CLR (Istruzione)

Azzera tutte le variabili numeriche mentre le variabili di stringa vengono inizializzate al valore "stringa nulla".
Le matrici vengono eliminate interamente annullando tutte le istruzioni DIM precedentemente eseguite.

Formalismo sintattico:

Esempio:

```
10 A = 1 : B = 2
20 PRINT A, B
30 CLR
40 PRINT A, B
```

Risultato:

LA LINEA 10 assegna alle variabili A e B i due valori specificati.
LA LINEA 20 stampa: 1   2.
LA LINEA 30 azzera le variabili A e B.
LA LINEA 40 stampa: 0   0.

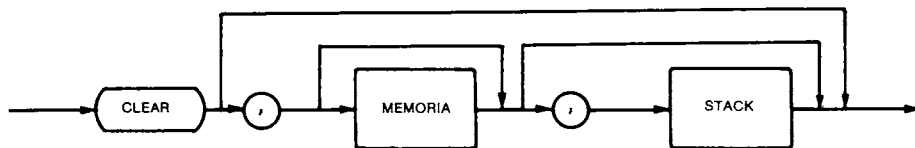
Note:
— Con CLR vengono anche annullate tutte le definizioni di funzioni fatte con la DEF FN.

— Non è possibile includere l'istruzione CLR all'interno di cicli FOR-NEXT o in sottoprogrammi BASIC.

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

CLEAR (Istruzione)

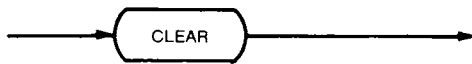
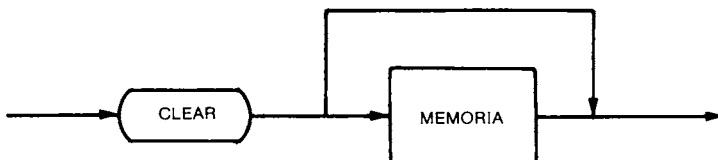
Azzera tutte le variabili numeriche, inizializza le variabili stringa al valore "stringa nulla" e chiude tutti i file dati. In taluni elaboratori è possibile, tramite l'istruzione CLEAR, stabilire lo spazio di memoria dedicato al BASIC e quello dedicato allo stack.

Formalismo sintattico:

Esempio:

```
10 A = 1 : B = 2
20 PRINT A, B
30 CLEAR
40 PRINT A, B
```

Risultato:

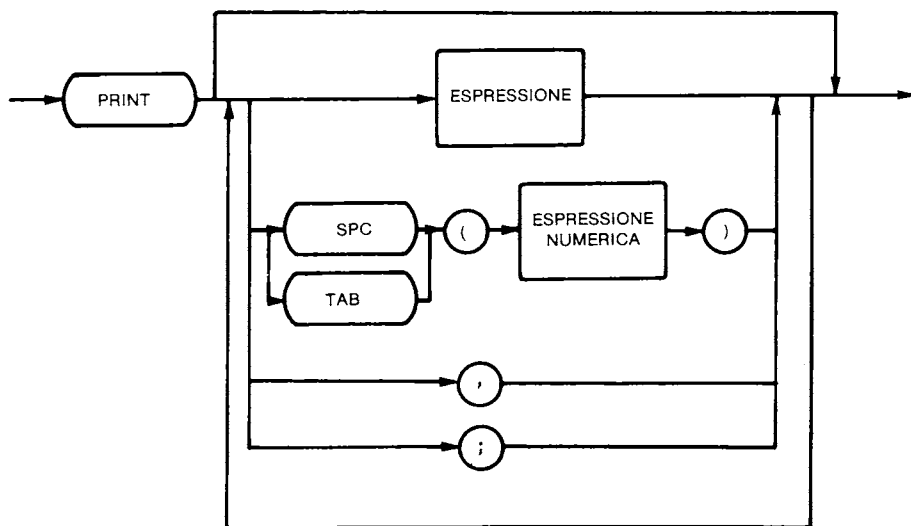
LA LINEA 10 assegna alle variabili A e B i due valori specificati.
LA LINEA 20 stampa: 1 2.
LA LINEA 30 azzera le variabili A e B.
LA LINEA 40 stampa: 0 0.

Variazione per: ZX 81
APPLE

Variazione per: ZX Spectrum

Note:
— Con CLEAR vengono annullate tutte le definizioni fatte con l'istruzione DEF (DEF FN, DEF SEG, DEF USR, ecc.).
— Non è possibile includere l'istruzione CLEAR all'interno di cicli FOR-NEXT o in sottoprogrammi BASIC.
— MEMORIA e STACK sono espressioni numeriche.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

PRINT (Istruzione)

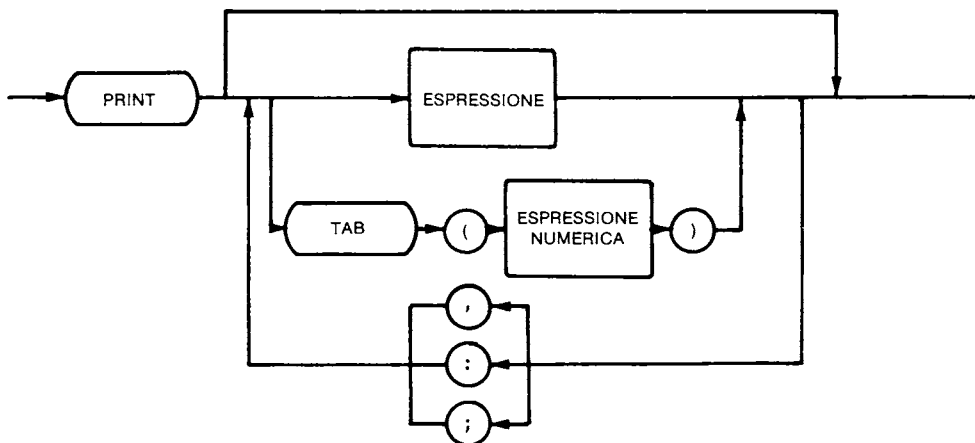
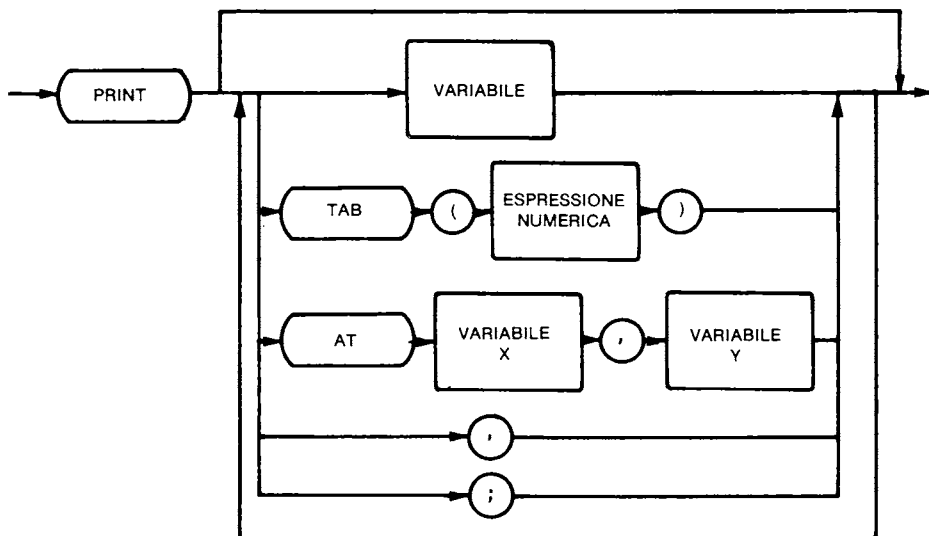
Consente di visualizzare i dati a video secondo un formato standard.

Formalismo sintattico:

Esempio:

```
10 A$ = "ABCD" : B$ = "EFGH" : C = 10
20 PRINT A$, B$
30 PRINT A$; B$
40 PRINT C; C
50 PRINT TAB (20); A$
60 PRINT SPC (20); A$
```

Risultato:

LA LINEA 10 assegna alle variabili stringa A$ e B$ rispettivamente ABCD e EFGH e alla variabile numerica C il valore 10.
LA LINEA 20 stampa: ABCD EFGH.
LA LINEA 30 stampa: ABCDEFGH.
LA LINEA 40 stampa: 10 10.
LA LINEA 50 stampa 19 spazi e poi la parola ABCD.
LA LINEA 60 stampa 20 spazi e poi la parola ABCD.

Variazione per: TI 99/4A

Variazione per: ZX 81
ZX Spectrum

Note:

— Se si usa un ; come separatore, il valore successivo viene stampato immediatamente dopo il precedente. Se si usa una virgola, il valore successivo viene stampato all'inizio della successiva zona di stampa (zone arbitrarie in cui viene suddiviso il video), che può anche essere alla riga successiva. Se si usa SPC, vengono inseriti tanti spazi quanto è il valore dell'espressione numerica.
Se si usa TAB, il successivo valore viene stampato alla colonna determinata dal valore dell'espressione numerica.
In entrambi i casi, ciò può far stampare i dati alla riga successiva.

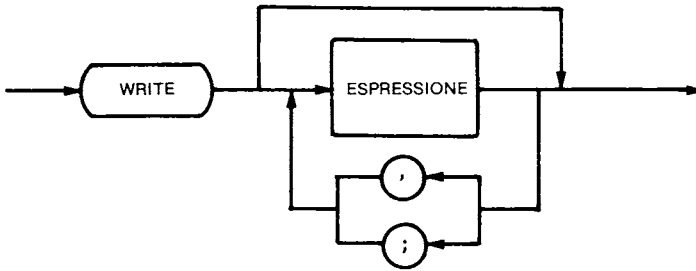
— Per lo ZX Spectrum è presente l'opzione AT. Questa permette di far stampare dati iniziando da una riga e una colonna specificate. Infatti il valore delle variabili X e Y determinano, rispettivamente, la colonna e la riga d'inizio della stampa dei successivi valori.

— Per stampare i dati con formato diverso, vedere PRINT USING.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

WRITE (Istruzione)

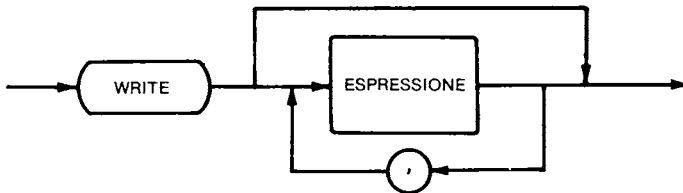
Emette i dati (presenti nella lista di espressioni) sullo schermo. Il BASIC esegue un ritorno a capo dopo la visualizzazione dell'ultimo dato.

Formalismo sintattico.

Esempio:

```
10 A$ = "ABCD" : B$ = "EFGH" : C = 10
20 WRITE A$, B$
30 WRITE C, C, A$
```

Risultato:

LA LINEA 10 permette di effettuare gli assegnamenti voluti alle variabili A$, B$, C.
LA LINEA 20 stampa a video i seguenti valori:
"ABCD", "EFGH"
LA LINEA 30 stampa a video i seguenti valori:
10, 10, "ABCD".

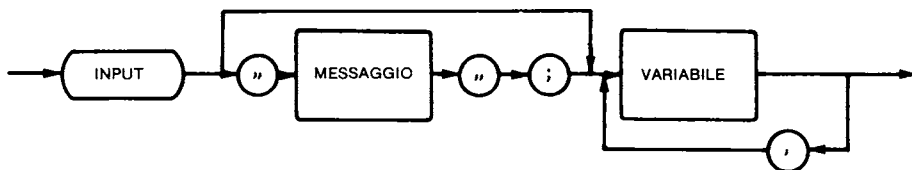
Variante per: M20

Note:

— La differenza tra WRITE e PRINT consiste nel fatto che WRITE inserisce delle virgole tra gli elementi visualizzati sullo schermo, e racchiude le stringhe tra apici all'atto della visualizzazione.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

INPUT (Istruzione)

Legge dati da tastiera e li assegna ad una o più variabili specificate, durante l'esecuzione del programma.

Formalismo sintattico:

Esempio:

```
10 REM SUBROUTINE INPUT
20 INPUT "NUMERO D'ORDINE:"; ORDER %
30 INPUT "NOME CLIENTE:"; NOME$
40 INPUT "DATA ORDINE:"; G, M, A
50 RETURN
```

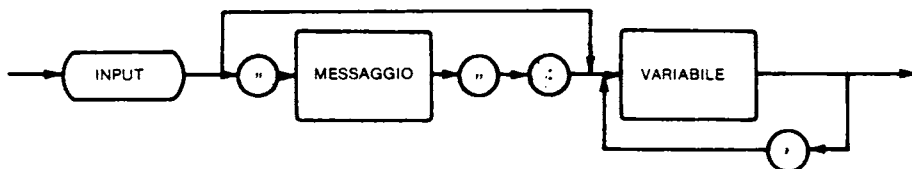
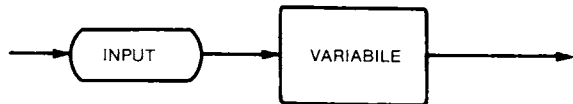
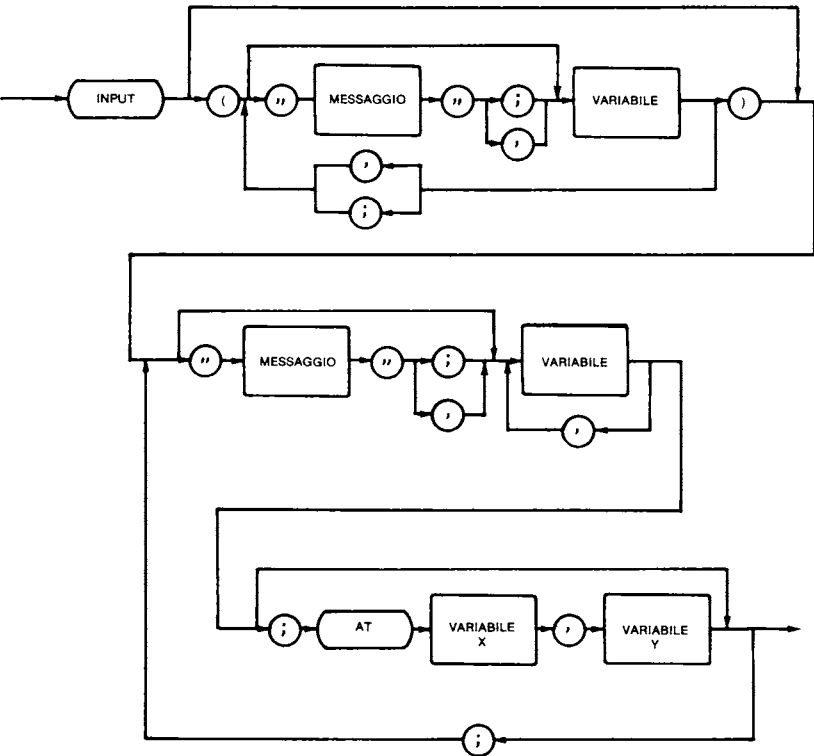
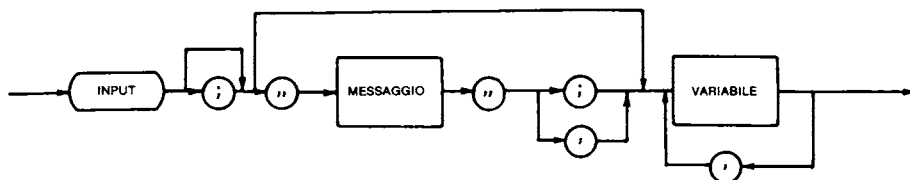
Risultati:

LINEA 10: Commenti per documentare la subroutine.
LINEA 20: Stampa la stringa e assegna il valore in ingresso alla variabile ORDER %.
LINEA 30: Stampa la stringa e assegna il valore stringa in ingresso alla variabile NOME$.
LINEA 40: Stampa la stringa e assegna i tre valori in ingresso, rispettivamente, alle variabili G, M ed A. I tre valori in ingresso devono essere numerici e separati tra loro da una virgola.
LINEA 50: Ritorna l'esecuzione al programma principale.

Variazione per: TI 99/4A

Variazione per: ZX 81

Variazione per: ZX Spectrum

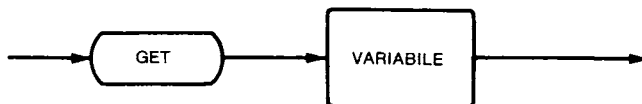
Variazione per: IBM P.C. - M20

Note:

— L'istruzione INPUT ferma l'esecuzione del programma mentre aspetta un input da tastiera.
— La digitazione del tasto RETURN indica la fine della composizione dell'input da tastiera.
— Un punto di domanda viene solitamente visualizzato alla sinistra del cursore. In certi sistemi tale punto di domanda può anche venir soppresso.
— Se si tenta di inserire un valore non numerico in una INPUT seguita da una variabile numerica, si verificherà un errore.

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

GET (Istruzione)

Legge un carattere dalla tastiera.

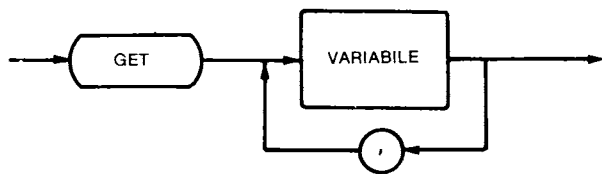
Formalismo sintattico:

Esempio:

```
100 GET A$ : IF A$ = " " THEN 100
110 IF A$ = "A" THEN GOSUB 200
120 END
    :
200 REM * SUBROUTINE 1 *
```

Risultato:

LA LINEA 100 assegna alla variabile A$ il carattere corrispondente al tasto che viene premuto: se il contenuto di A$ è uguale a spazi, vuol dire che non è stato digitato nessun tasto, per cui il controllo dell'esecuzione viene rimandato alla linea 100 stessa.
Questo è un "loop" che viene interrotto solo quando si preme un tasto della console.

LA LINEA 110 sposta il controllo dell'esecuzione alla linea 200 se è stato premuto il tasto A, altrimenti viene eseguita la linea 120 che fa terminare l'esecuzione del programma.

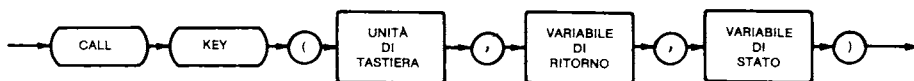
Variazione per: VIC 20
CBM 64

Note:

— L'istruzione GET (come la CALL KEY e la INKEY$) fa effettuare al sistema una esplorazione della tastiera, dopodichè, qualsiasi sia il risultato, l'esecuzione del programma riprende dalla istruzione seguente. Per dar quindi tempo all'utente di determinare quale tasto utilizzare e di effettuare l'operazione manuale di digitazione, è *indispensabile* corredare l'istruzione GET di un ciclo di ritardo (seconda istruzione della linea 100 dell'esempio).
— L'uso dell'istruzione GET consente di introdurre un singolo carattere senza interferire col video (il carattere non viene stampato) e senza utilizzare il tasto ENTER (o RETURN, o CR, ecc.).
— Se la variabile definita nella GET è numerica, il tentativo di introdurre valori non numerici darà origine ad errore.
— L'istruzione GET è presente con altri significati.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CALL KEY (Istruzione)

Trasferisce un carattere dalla tastiera all'interno del programma.

Formalismo sintattico:

Esempio:

```
100 CALL KEY (0, KEY, ST)
110 IF ST = 0 THEN 100
120 N = KEY – 64
130 PRINT N
```

Risultato:

LINEA 100: poichè l'unità di tastiera è zero, trasferisce il codice ASCII del carattere premuto nella variabile di ritorno KEY e posiziona la variabile di stato ST.

LINEA 110: se la variabile di stato è uguale a zero (nessun tasto premuto) sposta l'esecuzione di nuovo alla linea 100. Questo determina un "loop" di esecuzione, che termina quando si digita un tasto della console.

LINEE 120 e 130: utilizza il codice ASCII del carattere digitato per determinare la variabile N, che poi stampa.

Note:

— L'unità di tastiera è un'espressione numerica che può assumere i seguenti valori:
0 = tastiera della console (modo standard).
1 = parte sinistra della tastiera della console o controllo a distanza 1.
2 = parte destra della tastiera della console o controllo a distanza 2.
3, 4, 5 = modi particolari per la tastiera della console (standard, PASCAL, BASIC).

— La variabile di ritorno deve essere una variabile numerica.

— La variabile di stato è una variabile numerica che può assumere i seguenti valori:
+1 = è stato premuto un tasto diverso rispetto all'esecuzione della CALL KEY precedente;

—1 = è stato premuto lo stesso tasto che era stato premuto durante l'esecuzione della CALL KEY precedente.
0 = non è stato premuto nessun tasto.

— L'istruzione CALL KEY (come la GET e la INKEY$) fa effettuare al sistema una esplorazione della tastiera, dopodichè, qualsiasi sia il risultato, l'esecuzione del programma riprende dall'istruzione seguente. Per dar quindi tempo all'utente di determinare quale tasto utilizzare e di effettuare l'operazione manuale di digitazione, è *indispensabile* corredare l'istruzione CALL KEY di un ciclo di ritardo (alla linea 110 dell'esempio).
— L'uso dell'istruzione CALL KEY consente di introdurre un singolo carattere senza interferire col video (il carattere non viene stampato) e senza utilizzare il tasto ENTER.

INKEY$ (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Legge un carattere dalla tastiera.

Formalismo sintattico:

Esempio:

```
100 PRINT "CONTINUARE : C"
110 R$ = INKEY$ : IF R$ = " " THEN 110
120 IF R$ <> "C" THEN 110
130 REM * PROSEGUIMENTO *
```

Risultato:

LA LINEA 100 stampa il messaggio specificato.
LA LINEA 110 consente l'input di un carattere che viene assegnato alla variabile R$.
Se il contenuto di R$ è uguale a spazi, vuol dire che non è stato digitato nessun tasto, per cui il controllo dell'esecuzione viene riman-

dato alla linea 110 stessa. Questo è un "loop" che viene interrotto solamente quando si preme un tasto della console.

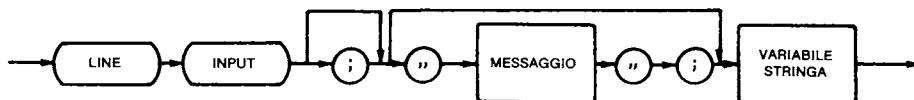
LA LINEA 120 sposta il controllo dell'esecuzione alla linea 110 se il tasto digitato non è il tasto C, altrimenti l'esecuzione prosegue in sequenza.

Note:
— La variabile che conterrà il risultato della funzione, deve essere una variabile stringa.
— L'uso della funzione INKEY$ consente di introdurre un singolo carattere senza interferire col video (il carattere non viene stampato) e senza utilizzare il tasto ENTER (o RETURN, o CR, ecc.).

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

LINE INPUT (Istruzione)

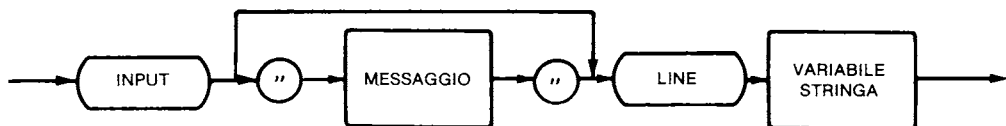
Permette l'introduzione di una intera linea, fino al ritorno a capo, e la assegna ad una variabile stringa senza far uso di delimitatori.

Formalismo sintattico:

Esempio:

```
110 PRINT "INTRODUCI NOME, COGNOME"
110 LINE INPUT L$
```

Risultato:

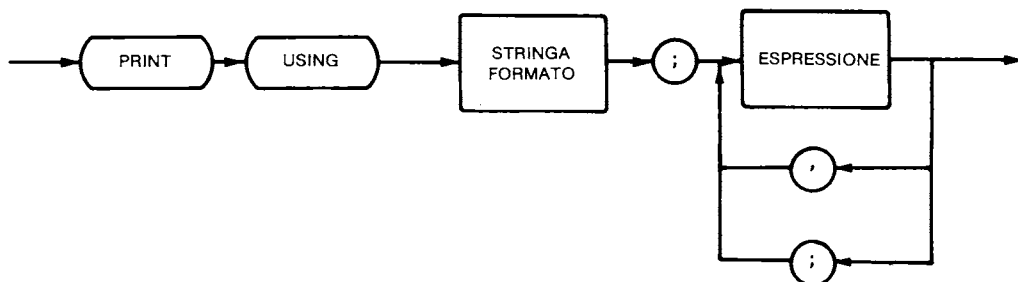
LA LINEA 100 stampa il messaggio sul video.
LA LINEA 110 prende l'input dalla tastiera e lo assegna alla variabile L$.

Variazione per: ZX Spectrum.

Note:
— Caratteri come la virgola ed il punto e virgola sono accettati come parte della stringa.
— I caratteri numerici fanno parte della stringa ma devono essere convertiti (ad esempio usando la funzione VAL).
— La linea introdotta può essere lunga fino a 254 caratteri.

PRINT USING (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Permette di visualizzare a video una lista di dati secondo un formato definito dall'utente.

Formalismo sintattico:

Esempio:

```
10 A$ = "TOPO" : B$ = "LINO"
20 PRINT USING "!"; A$; B$
30 PRINT USING "\ \ "; A$; B$,
40 PRINT USING "&"; B$
50 A = 15.7 : B = 9.6 : C = 55.669 : D = - .157
60 PRINT USING "# # . # #"; A; B; C; D
70 PRINT USING "# # . # # -"; A; B; C; D
80 PRINT USING "+ # # . # #"; A; B; C; D
```

Risultati:

LINEA 10: assegna alle variabili stringa A$ e B$ i valori specificati.

LINEA 20: stampa il primo carattere del contenuto delle variabili stringa, e cioè:
TL

LINEA 30: stampa i primi tre caratteri del contenuto delle variabili stringa, e cioè:
TOPLIN

LINEA 40: stampa l'intero contenuto di B$ nella seconda zona di stampa sulla stessa riga della stampa precedente:
TOPLIN LINO

LINEA 50: assegna i valori specificati alle variabili numeriche A, B, C, D.

LINEA 60: stampa i valori contenuti nelle variabili numeriche nel modo seguente:
15.70 9.60 55.67 0.157

LINEA 70: stampa i valori contenuti nelle variabili numeriche nel modo seguente:
15.70 9.60 55.67 0.157—

LINEA 80: stampa i valori contenuti nelle variabili numeriche nel modo seguente:
+15.70 +9.60 +55.67 —0.157

Note:

— I caratteri di formattazione utilizzabili all'interno della stringa formato sono parecchi e diversificati nei vari dialetti. I principali e piu usati sono i seguenti:

! stampa il primo carattere della stringa (per stringhe)

\n spazi \ specifica che devono essere stampati 2 + n caratteri della stringa (in MZ-700 si ha & n spazi &) (per stringhe)

& specifica un campo a stringa di lunghezza variabile (per stringhe)

rappresenta la posizione di ogni cifra da stampare (per campi numerici)

. punto decimale inseribile in qualsiasi posizione (per campi numerici)

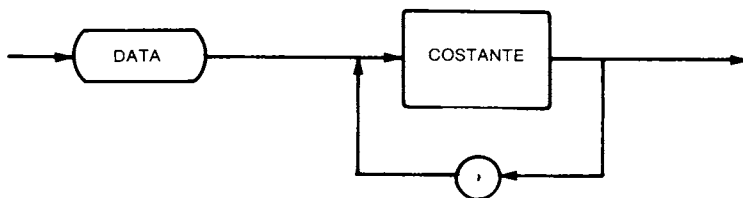
+	stampa il segno del numero nella posizione specificata (per campi numerici)
—	stampa il segno dopo l'ultima cifra significativa, se il numero è negativo (per campi numerici)
$ $	stampa il segno del dollaro alla sinistra del numero (per campi numerici)
^ ^ ^ ^	specificano il formato esponenziale per campi numerici. In MZ-700 sono sostituiti da ↑↑↑↑.

— Per la lista completa dei caratteri di formattazione, vedere l'allegato A, alla voce "STRINGA FORMATO".

DATA (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Crea un file di dati interno letto con istruzioni READ.

Formalismo sintattico:

Esempio:

```
100 READ X, Y, Z, A$
110 PRINT X, Y, Z, A$
120 DATA 5, 6
130 DATA 7, PIPPO
```

Risultato:

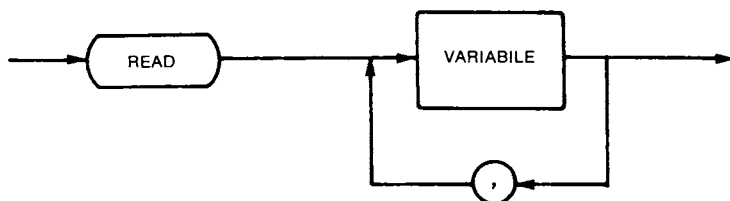
LA LINEA 100 assegna alle variabili X, Y, Z i valori contenuti nelle successive istruzioni DATA trovate. X, Y, Z assumono rispettivamente i valori 5, 6, 7, mentre A$ assume il valore PIPPO.

LA LINEA 110 stampa: 5 6 7 PIPPO.
LE LINEE 120 e 130 non vengono eseguite.

Note:
— Le istruzioni DATA possono essere messe in un punto qualsiasi del programma.
— Se ci sono più istruzioni DATA, esse vengono lette nell'ordine crescente del numero di linea.
— Il tipo di variabile che conterrà il valore letto in una istruzione DATA deve corrispondere al tipo di costante letta: quindi o entrambe numeriche o entrambe a stringa.
— Il numero di costanti presenti nelle istruzioni DATA deve essere uguale o maggiore al numero di variabili presenti nelle istruzioni READ, altrimenti si verifica un errore.
— Le istruzioni DATA vengono ignorate durante l'esecuzione, e vengono lette solo quando vengono eseguite delle istruzioni READ.

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

READ (Istruzione)

Legge i dati del file interno, creato con una o più istruzioni DATA, e li assegna alle variabili specificate.

Formalismo sintattico:

Esempio:

```
10 FOR I = 1 TO 5
20 READ A
30 PRINT A
40 NEXT I
50 DATA 1, 2, 3, 6, 4
```

Risultato:

LE LINEE dalla 10 alla 40 costituiscono un ciclo di lettura e stampa dei valori contenuti nell'istruzione DATA.

LA LINEA 20 legge un valore dalla prossima istruzione DATA incontrata e lo assegna alla variabile A.

La stampa ottenuta è:

1
2
3
6
4

Note:

— Le istruzioni READ e DATA lavorano sia con costanti e variabili numeriche sia con costanti e variabili stringa.

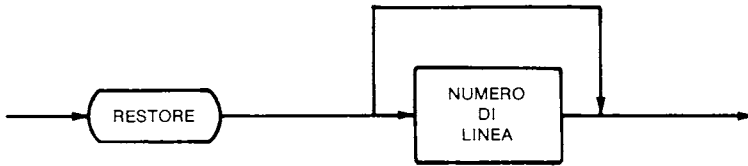
— Un'unica istruzione READ può accedere a più istruzioni DATA, oppure più istruzioni READ possono accedere alla stessa istruzione DATA.

Vedere anche l'istruzione DATA.

RESTORE (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
:	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Sposta il puntatore all'inizio di un file dati interno, creato da una istruzione DATA, o al numero di linea specificato (se presente).

Formalismo sintattico:

Esempio:

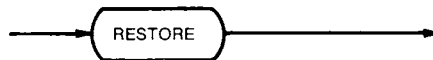
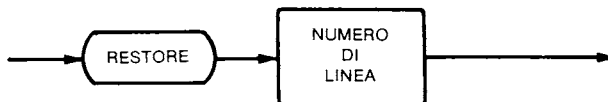
```
100 FOR I = 1 TO 2
110 RESTORE
120 FOR K = 1 TO 3
130 READ A
140 PRINT A
150 NEXT K
160 NEXT I
170 DATA 18, 200, 1
```

Risultato:

LE LINEE dalla 120 alla 150 costituiscono un ciclo di lettura e stampa dei valori contenuti nell'istruzione DATA.
Tale ciclo viene attivato 2 volte dalle linee 100 e 160.

LA LINEA 110 sposta il puntatore all'inizio dei valori dell'istruzione DATA.
La stampa ottenuta è:

```
18    200    1
18    200    1
```

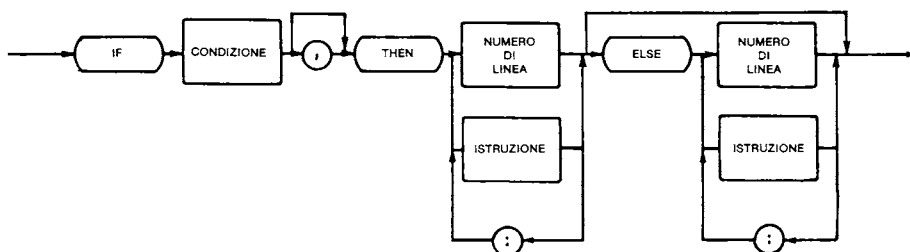
Variazione per: VIC 20
CMB 64
APPLE

Variazione per: ZX Spectrum

Note:

— NUMERO DI LINEA è una costante numerica che può assumere valori compresi fra 0 e 65.535.
Vedere anche istruzioni READ e DATA.

IF ... THEN (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

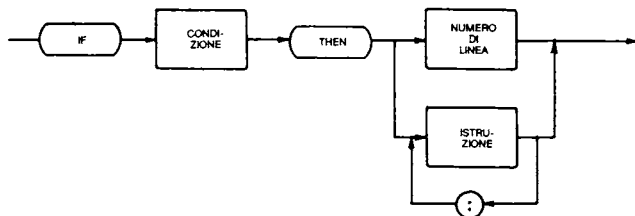
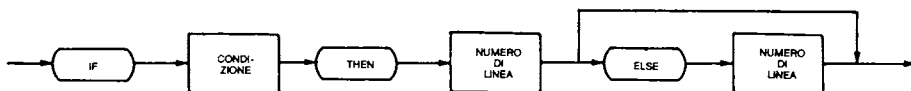
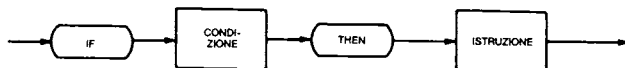
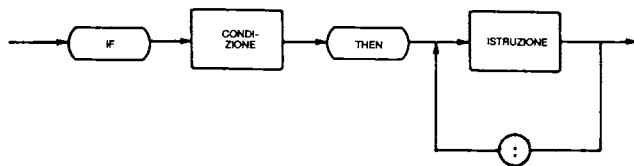
Effettua il trasferimento condizionato ad una istruzione specificata.

Formalismo sintattico:

Esempio:

```
 100 INPUT A$
 110 IF A$ = "S" THEN END
 120 IF A$ = "C" THEN GOSUB 1000
 130 PRINT A$ : GO TO 100
     :
1000 REM * SUBROUTINE 1 *
```

Risultato:

LA LINEA 100 assegna l'input alla variabile stringa A$.
LA LINEA 110 termina l'esecuzione del programma se il contenuto di A$ è S, in caso contrario passa ad eseguire la linea 120 successiva.
LA LINEA 120 trasferisce l'esecuzione del programma alla linea 1000 se il contenuto di A$ è C, altrimenti passa ad eseguire la linea 130 successiva.
LA LINEA 130 stampa il contenuto di A$ e sposta il controllo dell'esecuzione di nuovo alla linea 100.

Variazione per: VIC 20
CBM 64
MZ 700

Variazione per: TI 99/4A

Variazione per: ZX 81

Variazione per: ZX Spectrum
Apple

Note:

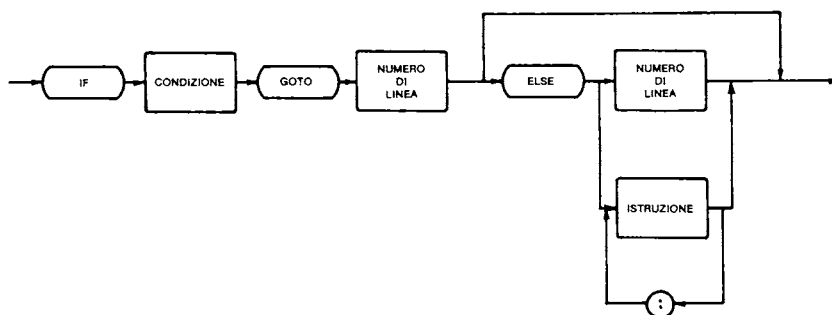
— NUMERO DI LINEA è una costante numerica che può assumere valori da zero a 65.535.

— ISTRUZIONE può essere qualunque istruzione BASIC vista, funzionante in modo differito e presente su quel particolare elaboratore su cui si sta eseguendo la IF.
Per CONDIZIONE vedere APPENDICE A.

IF ... GOTO (Istruzione)

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

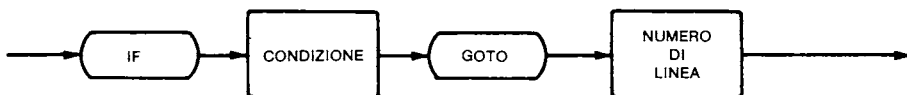
Effettua il trasferimento condizionato ad una istruzione specificata.

Formalismo sintattico:

Esempio:

```
100 INPUT A$
110 IF A$ = "A" GO TO 200
120 END
  :
200 PRINT "LINEA 200"
210 END
```

Risultato:

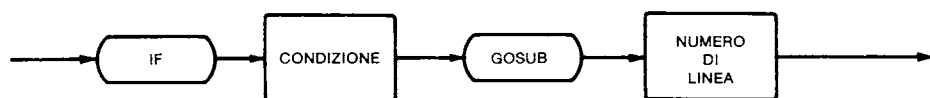
LA LINEA 100 assegna l'input alla variabile A$.
LA LINEA 110 trasferisce l'esecuzione del programma alla 200 se A$ contiene il valore A, altrimenti viene eseguita la linea 120 che fa terminare il programma.
ALLA LINEA 200 viene stampato il messaggio specificato, mentre la linea 210 fa terminare l'esecuzione del programma.

Variazione per: Apple
VIC 20
CBM 64
MZ 700

Note:
— In alcuni elaboratori può essere sostituito da:
IF <condizione> THEN GOTO <numero di linea>.
— NUMERO DI LINEA è una costante numerica che può assumere valori compresi fra 0 e 65535.

IF ... GOSUB (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Salta alla routine che comincia al numero di riga specificato se la condizione è soddisfatta.
L'ultima istruzione della subroutine richiamata deve essere una RETURN.

Formalismo sintattico:

Esempio:

```
100 INPUT A
110 IF A > 0 GOSUB 200
120 IF A = 0 GOSUB 400
130 IF A < 0 GOSUB 300
140 END
    :
200 REM * POSITIVO *
    :
290 RETURN
300 REM * NEGATIVO *
    :
390 RETURN
400 REM * UGUALE A ZERO *
    :
490 RETURN
```

Risultato:

LA LINEA 100 assegna l'input da tastiera alla variabile A.

ALLA LINEA 110, se il valore di A è positivo, viene spostato il controllo alla subroutine che inizia alla linea 200, altrimenti viene eseguita la successiva istruzione.

ALLA LINEA 120, se il valore di A è uguale a zero, viene spostato il controllo alla subroutine che inizia alla linea 400, altrimenti viene eseguita la successiva linea 130.

ALLA LINEA 130, se il valore di A è negativo, viene spostato il controllo alla subroutine che inizia alla linea 300, altrimenti viene eseguita la linea 140 che fa terminare il programma.

Note:

— NUMERO DI LINEA è una costante numerica che può assumere valori compresi fra 0 e 65535.

— In alcuni elaboratori può essere sostituito da:

IF <condizione> THEN GOSUB <numero di linea>.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

GOTO (Istruzione)

Trasferisce il controllo dell'esecuzione alla linea di programma specificata (salto incondizionato).

Formalismo sintattico:

Esempio:

```
 10 GO TO 100
 20 PRINT "LINEA 20"
 30 END
    :
100 PRINT "LINEA 100"
110 GO TO 20
```

Risultato:

LA LINEA 10 trasferisce l'esecuzione del programma alla linea 100.
LA LINEA 100 stampa:
LINEA 100
LA LINEA 110 trasferisce l'esecuzione del programma alla linea 20.
LA LINEA 20 stampa:
LINEA 20
LA LINEA 30 termina l'esecuzione del programma.

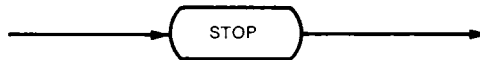
Note:

— NUMERO DI LINEA è una costante numerica che può assumere valori compresi fra 0 e 65.535.
— Si tende a preferire GOSUB a GOTO sia per avere programmi meglio strutturati sia perchè con GOSUB è possibile richiamare la subroutine più di una volta.
— Il numero di riga specificato in una istruzione GOTO non può essere quello di una linea inserita in un ciclo FOR-NEXT.
— Nei Sinclair ZX 81 e ZX Spectrum il numero di linea può essere una espressione numerica.

STOP (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Interrompe temporaneamente l'esecuzione del programma e riporta l'elaboratore al modo immediato (stato comandi).

Formalismo sintattico:

Esempio:

```
10 PRINT "INIZIO PROGRAMMA"
20 STOP
30 GO TO 50
   :
50 PRINT "FINE PROGRAMMA"
```

Risultato:

LA LINEA 10 stampa la stringa:
INIZIO PROGRAMMA

LA LINEA 20 blocca temporaneamente l'esecuzione del programma e viene stampato il messaggio: BREAK IN 20
Battendo CONT il programma prosegue con le istruzioni successive (30, 50...).

Note:

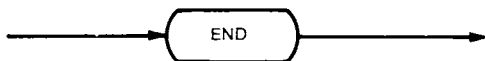
— Diversamente dall'istruzione END, l'istruzione STOP non chiude i file.

— È possibile utilizzare l'istruzione STOP per controllare, in fase di prova, la corretta esecuzione del programma. Infatti, mentre il programma è temporaneamente fermo, è possibile usare comandi o istruzioni (ad esempio la PRINT) in modo immediato per verificare i risultati parziali. È possibile poi far ripartire il programma con il comando CONTINUE.

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

END (Istruzione)

Fa terminare l'esecuzione del programma, chiude tutti i file dati e riporta l'elaboratore al modo immediato (stato comandi).

Formalismo sintattico:

Esempio:

```
 10 PRINT "FINE LAVORO = Y"
 20 INPUT A$
 30 IF A$ = "Y" THEN 100
    :
100 END
```

Risultato:

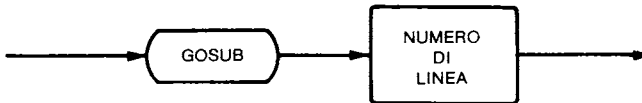
LA LINEA 10 stampa il messaggio specificato.
LA LINEA 20 assegna l'input da tastiera alla variabile A$. Alla linea 30 se il contenuto di A$ è uguale a Y, il controllo viene spostato alla linea 100, altrimenti viene eseguita l'istruzione successiva. La linea 100 fa terminare l'esecuzione del programma.

Note:
— Alcuni BASIC non richiedono l'istruzione END al termine del programma.
— L'istruzione END, a differenza della STOP, non fa comparire nessun messaggio.
— Non è possibile, dopo l'esecuzione di una istruzione END, far proseguire l'esecuzione del programma con l'utilizzo del comando CONTINUE, anche se dopo la END sono presenti altre istruzioni.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

GOSUB (Istruzione)

Richiama una "subroutine" passando il controllo al suo primo numero di linea.
La subroutine deve terminare con una istruzione RETURN.

Formalismo sintattico:

Esempio:

```
 100 PRINT "MENU"
 110 PRINT "(1) CARICAMENTO"
 120 PRINT "(2) VARIAZIONE"
 130 PRINT "(3) STAMPE"
 140 INPUT "SCELTA:"; A$
 150 IF A$ = "1" THEN GOSUB 800
 160 IF A$ = "2" THEN GOSUB 900
 170 IF A$ = "3" THEN GOSUB 1000
      :
 800 REM * SUBROUTINE CARICAMENTO *
 890 RETURN
 900 REM * SUBROUTINE VARIAZIONE *
 990 RETURN
1000 REM * SUBROUTINE STAMPE *
1090 RETURN
```

Risultato:

LE LINEE 100-130 stampano il menu.
LA LINEA 140 consente di effettuare la scelta.
LE LINEE 150-170 trasferiscono l'esecuzione del programma alle linee 800, 500, 1000 a seconda del valore di A$.

Note:
— Una subroutine deve sempre terminare con RETURN.
— L'istruzione RETURN causa il proseguimento dell'esecuzione del programma con la prossima istruzione dopo il GOSUB che ha richiamato la subroutine.

— NUMERO DI LINEA è una costante numerica che può assumere valori compresi tra 0 e 65.535.
— È possibile utilizzare "annidamenti" di subroutine, cioè subroutine inserite in altre subroutine a loro volta inserite in subroutine a livello più alto, e così via. L'importante è che le subroutine più interne vengano completamente eseguite prima del termine delle subroutine esterne.

Esempio di "annidamento" corretto:

```
  10 REM * MAIN *
     :
  50 GOSUB 100
     :
  90 END
 100 REM * SUB. LIVELLO 1 *
     :
 200 GOSUB 1000
     :
 990 RETURN
1000 REM * SUB. LIVELLO 2 *
     :
1300 GOSUB 2000
     :
1990 RETURN
2000 REM * SUB. LIVELLO 3 *
     :
2990 RETURN
```

Esempio di "annidamento" errato.

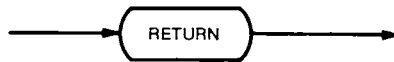
```
  10 REM * MAIN *
     :
  50 GOSUB 100
     :
  90 END
 100 REM * SUB. LIVELLO 1 *
     :
 200 GOSUB 100
     :
 990 RETURN
```

Non esiste limite al numero di livelli di annidamento, se non quello dovuto alle limitazioni di memoria del sistema.
— Nei Sinclair ZX 81 e ZX Spectrum il numero di linea è una espressione numerica.

RETURN (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Trasferisce il controllo alla prima istruzione successiva all'ultima GOSUB (oppure ON ... GOSUB) eseguita.

Formalismo sintattico:

Esempio:

```
 100 GOSUB 1000
 110 PRINT "FINE PROGRAMMA"
 120 END
     :
1000 PRINT "ESECUZIONE SUBROUTINE"
     :
1100 RETURN
```

Risultato:

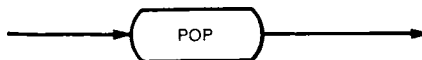
LA LINEA 100 richiama la subroutine che inizia alla linea 1000.
LA LINEA 1000 stampa: ESECUZIONE SUBROUTINE.
LA LINEA 1100 ritorna l'esecuzione alla 110.
LA LINEA 110 stampa: FINE PROGRAMMA.

Note:
— Una subroutine deve sempre terminare con RETURN.
— Per maggiori informazioni consultare le istruzioni GOSUB e ON ... GOSUB.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

POP (Istruzione)

L'istruzione POP ha l'effetto di un RETURN ma senza il salto. Il successivo RETURN incontrato, anzichè saltare ad una istruzione oltre il GOSUB più di recente eseguito, salta ad una istruzione oltre il *secondo* GOSUB più di recente eseguito (cioè il penultimo GOSUB eseguito).

Formalismo sintattico:

Esempio:

```
100 GOSUB 200
110 PRINT "PRIMA SUBROUTINE"
120 END
    :
200 GOSUB 300
210 PRINT "SECONDA SUBROUTINE"
    :
300 POP
310 RETURN
```

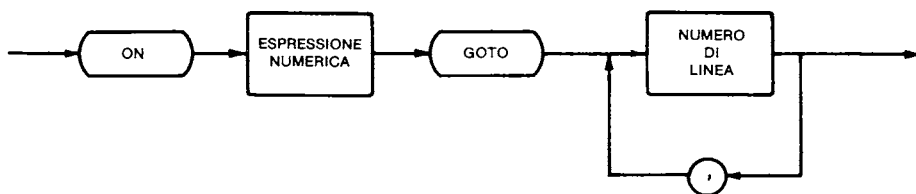
Risultato:

LA LINEA 100 salta alla 200.
LA LINEA 200 salta alla 300.
LA LINEA 300 predispone l'area di stack per il ritorno all'istruzione successiva il penultimo GOSUB eseguito.
LA LINEA 310 ritorna quindi alla 110.
LA LINEA 110 stampa: PRIMA SUBROUTINE.

ON ... GOTO (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Trasferisce il controllo dell'esecuzione ad una linea scelta tra un insieme di linee specificate dopo GOTO, in funzione del valore assunto da un'espressione specificata dopo ON.

Formalismo sintattico:

Esempio:

```
 10 PRINT "MENU"
 20 PRINT "1) CARICAMENTO"
 30 PRINT "2) VARIAZIONI"
 40 INPUT "SCELTA:"; K
 50 ON K GOTO 300, 400
 60 END
  :
300 REM * CARICAMENTO *
400 REM * VARIAZIONI *
```

Risultato:

LE LINEE dalla 10 alla 30 visualizzano un menù.
LA LINEA 40 assegna la risposta alla variabile K.
LA LINEA 50 se K vale 1 fa saltare il programma alla linea 300; se K vale 2 fa saltare il programma alla linea 400; altrimenti prosegue in sequenza, eseguendo la linea 60 che fa terminare l'esecuzione del programma.

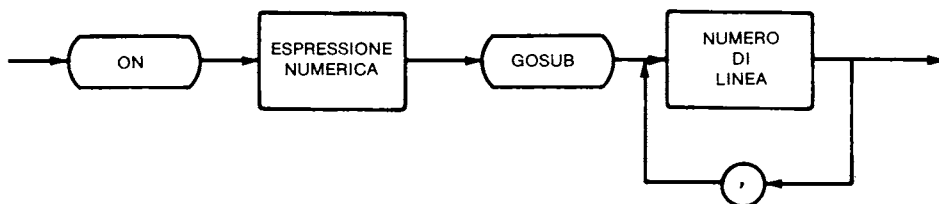
Note:
— Il valore dell'espressione numerica non deve essere superiore a 255.
— NUMERO DI LINEA è una costante numerica che può assumere valori compresi fra 0 e 65535.

— Se il valore dell'espressione numerica è 0 oppure è maggiore del numero di strade previste, l'esecuzione prosegue, in sequenza, con l'istruzione immediatamente seguente la ON ... GOTO.
— Se il valore dell'espressione numerica non è un intero, viene effettuato un troncamento della parte decimale prima di valutare l'espressione. In alcuni elaboratori si effettua invece un arrotondamento.

X	TI 99/4A
X	VIC 20
X	CBM 64
.	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

ON ... GOSUB (Istruzione)

Richiama una "subroutine" scelta tra n subroutine specificate, in funzione del valore assunto da una espressione specificata dopo ON.

L'ultima istruzione della subroutine deve essere una RETURN.

Formalismo sintattico:

Esempio:

```
 10 PRINT "MENU"
 20 PRINT "1) CARICAMENTO"
 30 PRINT "2) VARIAZIONI"
 40 INPUT "SCELTA:"; K
 50 ON K GOSUB 300, 400
 60 END
  :
300 REM * SUBROUTINE CARICAMENTO *
390 RETURN
400 REM * SUBROUTINE VARIAZIONI *
490 RETURN
```

Risultato:

LE LINEE	dalla 10 alla 30 visualizzano un menù.
LA LINEA	40 assegna la risposta alla variabile K.
LA LINEA	50 se K vale 1 cede il controllo alla subroutine che inizia alla linea 300; se K vale 2 cede il controllo alla subroutine che inizia alla linea 400; altrimenti prosegue in sequenza, eseguendo la linea 60 che fa terminare l'esecuzione.

Note:

— NUMERO DI LINEA è una costante numerica che può assumere valori compresi tra 0 e 65.535.

— Se il valore dell'espressione numerica è 0 oppure è maggiore del numero di strade previste, l'esecuzione prosegue, in sequenza, con l'istruzione immediatamene seguente la ON ... GOSUB.

— Se il valore dell'espressione numerica non è un intero, viene effettuato un troncamento della parte decimale prima di valutare l'espressione. In alcuni elaboratori si effettua invece un arrotondamento.

— Il valore dell'espressione numerica non deve essere superiore a 255.

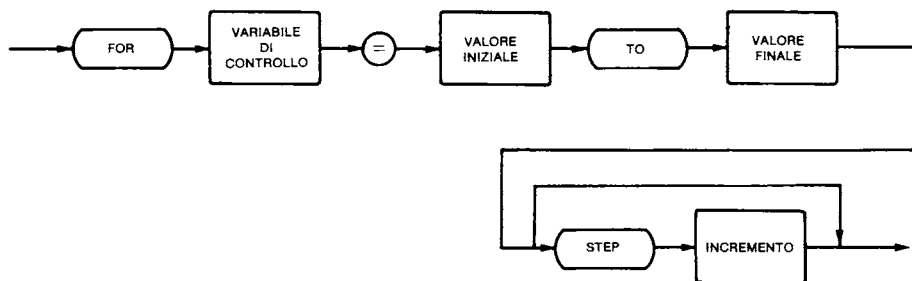
— Per ulteriori informazioni consultare l'istruzione GOSUB.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

FOR (Istruzione)

Consente di eseguire in "loop" una serie di istruzioni per un certo numero di volte.

Al termine delle istruzioni del ciclo deve essere presente una istruzione NEXT.

Formalismo sintattico

Esempio:

```
 90 N = 5
100 FOR I = 1 TO N - 1
110 READ X
120 PRINT X
130 NEXT I
140 DATA 1, 2, 3, 4
```

Risultato:

LA LINEA 90 assegna il valore 5 alla variabile N.
LA LINEA 100 innesca un ciclo da ripetersi 4 volte, incrementando la variabile I di 1, partendo dal valore 1.
LA LINEA 110 assegna uno dei valori dell'istruzione DATA alla variabile X.
LA LINEA 120 stampa il valore contenuto in X.
LA LINEA 130 chiude le istruzioni del ciclo e ritorna l'esecuzione del programma alla linea 100.

La stampa ottenuta è:

1
2
3
4

Note:

— VARIABILE DI CONTROLLO deve essere una variabile numerica.

— VALORE INIZIALE, VALORE FINALE e INCREMENTO sono espressioni numeriche.

— I cicli FOR-NEXT possono essere "annidati" (NESTED), cioè interni uno all'altro. In tal caso è bene rispettare alcune regole, per evitare che, all'atto dell'esecuzione, i risultati siano imprevedibili:
1) ad ogni istruzione FOR deve corrispondere una istruzione NEXT.
2) Per ogni ciclo deve essere usata una variabile di controllo diversa.
3) Ogni ciclo deve contenere completamente un ciclo di livello inferiore.
4) Ogni ciclo deve essere contenuto completamente in un ciclo di livello superiore.

Esempio di "annidamento" corretto:

```
 10 FOR I = 1 TO 10
 20 FOR J = 3 TO 7
 30 PRINT I, J
 40 FOR K = 0 TO 5
 50 PRINT (I * J) + K
 60 NEXT K
 70 PRINT "LIVELLO 2"
 80 NEXT J
 90 PRINT "LIVELLO 1"
100 NEXT I
```

Esempio di "annidamento" errato:

```
10 FOR I = 1 TO 10
20 FOR J = 3 TO 7
30 PRINT I * J
40 NEXT I
50 NEXT J
```

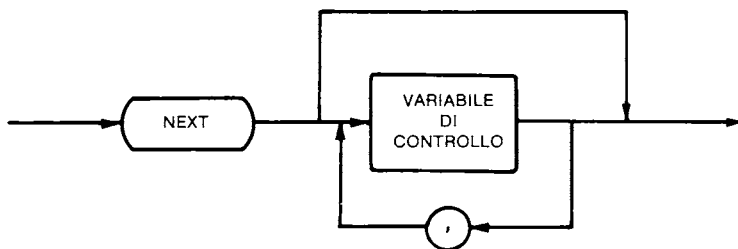
Non esiste limite al numero di livelli di annidamento, se non quello dovuto alle limitazioni di memoria del sistema.

— Alcune istruzioni (per esempio la CLR) non possono essere utilizzate all'interno di cicli FOR-NEXT.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

NEXT (Istruzione)

Chiude l'elenco delle istruzioni da eseguire nel ciclo iniziato con un'istruzione FOR.

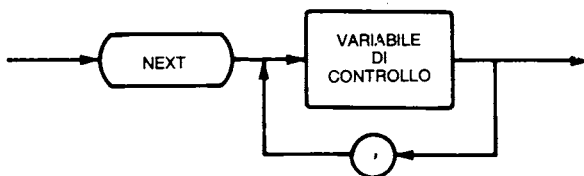
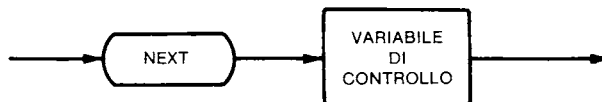
Formalismo sintattico:

Esempio:

```
100 FOR I = 1 TO 3
110 PRINT "***"
120 NEXT I
```

Risultato:

LA LINEA 100 attiva un ciclo da ripetersi 3 volte la cui variabile di controllo è I.
LA LINEA 110 effettua una stampa di ***.
LA LINEA 120 indica la fine delle istruzioni del ciclo relativo alla variabile I e ritorna l'esecuzione del programma alla linea 100.
La stampa ottenuta è:

```
***
***
***
```

Variazione per: TI 99/4A

Variazione per: ZX 81
ZX Spectrum

Note:

— VARIABILE DI CONTROLLO è una variabile numerica e deve essere la stessa variabile indicata nell'istruzione FOR a cui l'istruzione NEXT fa riferimento.

— Quando viene eseguita l'istruzione NEXT, viene aggiunto il valore presente in INCREMENTO (oppure il valore 1) alla variabile di controllo, e viene verificato se il nuovo valore della variabile di controllo supera il valore presente in VALORE FINALE: se si, il ciclo viene terminato, altrimenti viene eseguita una nuova iterazione.

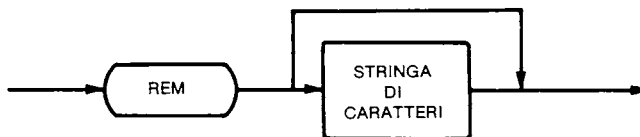
— Per ulteriori informazioni consultare l'istruzione FOR.

REM (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Serve per inserire commenti documentativi nel programma.

Non viene eseguita.

Formalismo sintattico:

Esempio:

```
10 REM PROGRAMMA DI CARICAMENTO
20 REM VERSIONE 1.1
30 REM 10 GENNAIO 1980
40 A = 1
50 B = 1
60 :
   :
   :
```

Risultato:

LE LINEE 10-30 non vengono eseguite, sono utilizzate per commentare il programma.

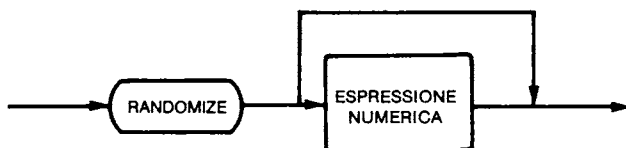
Note:

— La stringa di caratteri non necessita dei doppi apici.
— Ogni istruzione o funzione che segue una REM sulla stessa linea, non viene eseguita.
— È possibile (tranne che nel TI99/4A) aggiungere dei commenti alla fine di una riga. Per esempio:
  10 C = 2 * PI * R : REM Calcolo circonferenza
— In alcuni elaboratori è possibile sostituire : REM alla fine di una riga con un apostrofo. Esempio:
  10 C = 2 * PI * R ' Calcolo circonferenza.

RANDOMIZE (Istruzione)

X	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

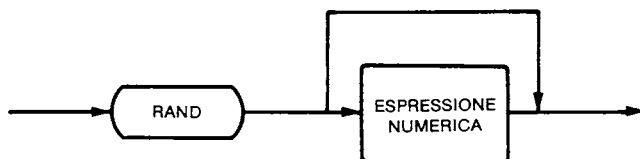
Modifica il generatore di numeri casuali (vedere funzione RND).

Formalismo sintattico:

Esempio:

```
100 RANDOMIZE
110 PRINT RND
120 END
```

Risultato:

LA LINEA 100 modifica il generatore di numeri casuali.
LA LINEA 110 stampa il numero casuale generato compreso tra 0 ed 1. Ad esempio: .135742.
Una successiva esecuzione del programma permetterà di generare un numero diverso da .135742.

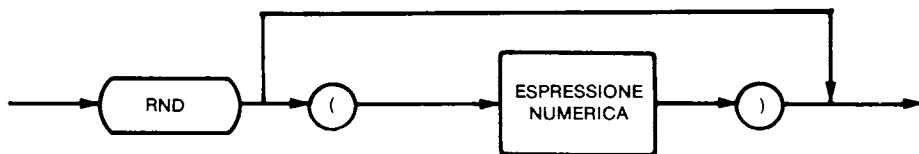
Variazione per: ZX 81

Note:

— L'istruzione RANDOMIZE è indispensabile, quando la funzione RND non preveda (o venga usata senza) l'espressione numerica, qualora si voglia modificare la sequenza dei numeri casuali generata ad ogni esecuzione del programma. Negli elaboratori in cui la funzione RND è necessariamente accompagnata da una espressione numerica, i valori assunti dall'espressione numerica stessa sono in grado di modificare il generatore di numeri casuali, permettendo di generare sequenze di numeri casuali sempre diverse ad ogni esecuzione del programma e rendendo quindi inutile l'istruzione RANDOMIZE (che infatti non è presente nell'interprete BASIC di tali elaboratori).

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

RND (Funzione)

Calcola un numero casuale compreso tra 0 e 1 (≥ 0 e $<$ 1). La stessa sequenza di numeri casuali viene generata ogni volta che il programma viene eseguito, a meno che il generatore dei numeri casuali venga reinizializzato (vedere l'istruzione RANDOMIZE).

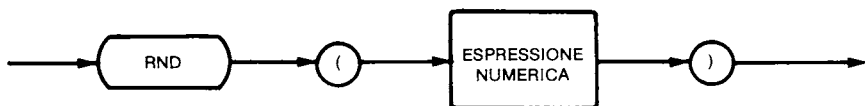
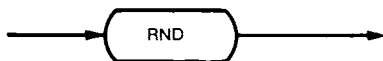
Formalismo sintattico:

Esempio:

```
100 RANDOMIZE
110 FOR I = 1 TO 5
120 PRINT RND (0)
130 NEXT I
140 END
```

Risultato:

LA LINEA 100 modifica il generatore di numeri casuali.
LA LINEA 110, unitamente alla 120, innesca un ciclo da ripetersi 5 volte, in cui la linea 120 stampa i numeri casuali generati, compresi tra 0 ed 1. Un esempio tipico può essere:

.135742
.243618
.510791
.304653
.971064

Variazione per: VIC 20
CBM 64
ZX 81
MZ 700
Apple

Variazione per: TI 99/4A
ZX Spectrum

Note:

— L'espressione numerica viene utilizzata per fare in modo che, ad ogni esecuzione del programma, venga generata una sequenza di numeri casuali sempre diversa dalla precedente. Per gli elaboratori in cui la RND è sprovvista di (o può essere utilizzata senza) espressione numerica, è possibile utilizzare l'istruzione RANDOMIZE.

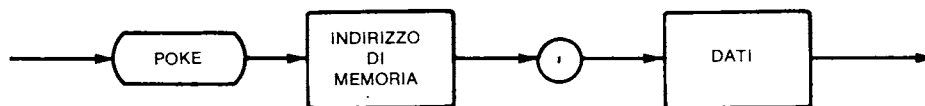
— La funzione RND genera un numero casuale compreso nell'intervallo 0 ÷ 1. Se si vuole generare un numero casuale compreso tra X e Y (con X ⩽ Y) si dovrà utilizzare la formula:

INT ((Y-X + 1) * RND) + X

	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ 700
X	IBM P.C.
	OLIVETTI M20
X	APPLE

POKE (Istruzione)

Scrive un byte in una posizione di memoria.

Formalismo sintattico:

Esempio:

```
10 POKE 10000, 12
20 POKE 10001, 53
30 A = PEEK (10000)
40 B = PEEK (10001)
50 PRINT A, B
```

Risultato:

LA LINEA 10 mette il valore 12 all'indirizzo 10000.
LA LINEA 20 mette il valore 53 all'indirizzo 10001.
LA LINEA 30 prende il contenuto dell'indirizzo 10000 e lo assegna alla variabile A.
LA LINEA 40 prende il contenuto dell'indirizzo 10001 e lo assegna alla variabile B.
LA LINEA 50 stampa: 12 53

Note:

— La funzione complementare dell'istruzione POKE è PEEK.
— L'istruzione POKE è frequentemente usata, assieme alle istruzioni READ e DATA, per memorizzare subroutine in linguaggio macchina generate all'interno di programmi in linguaggio BASIC.
— La scrittura di valori particolari in indirizzi prescelti di memoria è talvolta usata per eseguire funzioni particolari.
— L'interprete BASIC non controlla l'indirizzo di memoria, per cui bisogna utilizzare l'istruzione POKE con attenzione, per non modificare indirizzi importanti (aree di variabili, l'interprete BASIC stesso, ecc.).

MODO "VIRGOLETTE"

In alcuni elaboratori (VIC 20, CBM 64, MZ 700) è possibile, in concomitanza con l'istruzione PRINT, inserire (dopo aver aperto le virgolette e prima di chiuderle) alcuni caratteri speciali con funzioni ben determinate (movimento del cursore, controlli del colore, ecc.). Naturalmente è sempre possibile, in tale modo di funzionamento, inserire tutti gli altri caratteri della tastiera, compresi quelli della grafica.

VIC 20 e CBM 64

Movimento del cursore

Dopo aver aperto le virgolette, i tasti per il controllo del cursore non funzionano più in modo immediato, ma originano a video i seguenti caratteri speciali:

Tasto	Visualizzazione	Funzione
CLR/HOME		Sposta il cursore in alto a sinistra senza cancellare il video.
SHIFT CLR/HOME		Sposta il cursore in alto a sinistra dopo aver cancellato il video.
⇑ CRSR ⇓		Sposta il cursore di una posizione verso il basso.
SHIFT ⇑ CRSR ⇓		Sposta il cursore di una posizione verso l'alto.
⇐ CRSR ⇒		Sposta il cursore di una posizione a destra.
SHIFT ⇐ CRSR ⇒		Sposta il cursore di una posizione a sinistra.

Il movimento effettivo del cursore avverrà all'atto dell'esecuzione dell'istruzione PRINT contenente i caratteri speciali ora visti.

Se si vuole muovere il cursore in modo immediato, sarà necessario uscire dal modo virgolette, chiudendo le virgolette che erano state aperte in precedenza.

Caratteri "Reverse"

Tasto	Visualizzazione	Funzione
CTRL 9		Inizia la funzione "Reverse", per cui i caratteri successivi verranno stampati (all'atto dell'esecuzione) in VIDEO REVERSE, cioè come il negativo di un disegno.
CTRL 0		Termina la funzione di VIDEO REVERSE.

Modo inserimento

È un ulteriore modo di lavorare all'interno del modo virgolette, e viene impostato utilizzando il tasto INST DEL .

In tale modo di funzionamento i controlli cursore e i controlli colore appaiono come caratteri "Reverse".
Il tasto INST funziona ora come vero e proprio inserimento di spazi.

La condizione "modo inserimento" termina quando viene premuto il tasto RETURN.

Controlli del colore

Tasto	Visualizzazione	Colore
CTRL 1		Nero
CTRL 2		Bianco
CTRL 3		Rosso
CTRL 4		Azzurro
CTRL 5		Porpora
CTRL 6		Verde
CTRL 7		Blu
CTRL 8		Giallo
C= 1		Arancio
C= 2		Marrone
C= 3		Rosso chiaro
C= 4		Grigio 1

		Grigio 2
		Verde chiaro
		Blu chiaro
		Grigio 3

Tutti i caratteri che seguono questi caratteri speciali, verranno stampati, all'atto dell'esecuzione, col colore corrispondente.

SHARP MZ 700

Nel "Modo virgolette" è possibile inserire caratteri speciali per il controllo del cursore.

Per immettere tali caratteri è necessario dapprima premere il tasto GRAPH. Alla fine della digitazione, per ritornare ad immettere caratteri alfanumerici normali, premere il tasto ALPHA.

Tasto	Visualizzazione	Funzione
CLR	C	Cancella l'intero schermo e sposta il cursore in alto a sinistra.
HOME	H	Sposta il cursore in alto a sinistra senza cancellare lo schermo.
→	→	Sposta il cursore di una colonna a destra.
←	←	Sposta il cursore di una colonna a sinistra.
↑	↑	Sposta il cursore di una riga verso l'alto.
↓	↓	Sposta il cursore di una riga verso il basso.

Lo spostamento effettivo del cursore avverrà, naturalmente, all'atto dell'esecuzione dell'istruzione PRINT contenente i caratteri speciali ora visti.

CAPITOLO 3

RITORNO AL BASIC

GENERALITA'
BYE
SYSTEM

GENERALITA'

In qualunque momento si voglia lasciare l'esecuzione del programma in corso e ritornare all'interprete BASIC, è possibile utilizzare opportuni tasti.

Per:	TEXAS TI 99/4A	Tasti FCTN e = (QUIT)
Per:	VIC 20 CBM 64	Tasti RUN/STOP e RESTORE
Per:	MZ 700	Tasti CTRL e RESET
Per:	IBM P.C.	Tasti CTRL e BREAK
Per:	Olivetti M20	Tasti SHIFT e RESET
Per:	APPLE II	Tasti CTRL e C

Non esiste tale possibilità per: ZX 81 e ZX Spectrum.

BYE (Comando)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Serve per uscire dall'ambiente BASIC e ridare il controllo al Sistema Operativo. Chiude tutti i file aperti, cancella il programma e tutte le variabili.

Formalismo sintattico:

Esempio: (in modo immediato):

>BYE ENTER

Risultato:

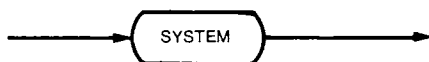
Nel TI 99/4A riporta il controllo dell'elaboratore alla maschera iniziale.
Nel MZ-700 riporta il controllo dell'elaboratore al programma Monitor della RAM (cioè quello presente nell'interprete BASIC).

Note:
— Nel TI 99/4A l'eventuale programma BASIC presente in memoria viene perso in modo definitivo.
— Nello SHARP MZ 700 l'eventuale programma BASIC presente in memoria non viene perso con l'esecuzione di un comando BYE. Basta infatti eseguire il comando monitor *R per ritornare in ambiente BASIC, col programma intatto.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

SYSTEM (Comando)

Consente di passare dall'interprete BASIC al Sistema Operativo, chiudendo tutti i file dati.
Il programma BASIC residente in memoria viene perso.

Formalismo sintattico:

Esempio:

```
100 PRINT "BATTI F PER FINE LAVORO"
110 INPUT A$
120 IF A$ = "F" THEN GO TO 200
    :
200 SYSTEM
```

Risultato:

LA LINEA 100 fornisce un messaggio.
LA LINEA 110 accetta una risposta e la assegna alla variabile A$.
LA LINEA 120 se A$ contiene F salta alla 200.
LA LINEA 200 chiude tutti i file e ritorna il controllo al sistema operativo.

Note:
— Il comando SYSTEM può essere usato sia in modo immediato che in modo differito.

CAPITOLO 4

CANCELLAZIONE VIDEO

GENERALITA'
CALL CLEAR
CLS
HOME

GENERALITA'

Il video può essere cancellato (sia in modo immediato che in modo programma), senza interferire sul contenuto della memoria, utilizzando opportuni tasti oppure opportuni comandi o istruzioni.

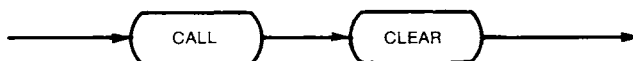
Per:	TEXAS TI 99/4A	CALL CLEAR
Per:	VIC 20 CBM 64	Tasto CLR oppure PRINT "♥" (vedere "Modo virgolette")
Per:	ZX 81 ZX Spectrum	CLS
Per:	MZ 700	Tasto CLR oppure PRINT "C" (vedere "Modo virgolette")
Per:	IBM P.C.	Tasti CTRL e HOME oppure CLS
Per:	Olivetti M20	CLS
Per:	Apple	Tasti "ESC + @" oppure CALL - 936 oppure HOME

CALL CLEAR (Comando-istruzione)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Viene usato dal TEXAS TI 99/4A per cancellare il video.

Riempie tutte le posizioni del video con spazi (cod. ASCII 32).

Formalismo sintattico:

Esempio:

In modo immediato:

> CALL CLEAR [ENTER]

In modo differito:

```
 100 CALL CLEAR
 110 GOSUB 1000
 120 CALL CLEAR
      :
1000 REM * SUBROUTINE DI STAMPA *
1010 PRINT "VALORE 1"; A
1020 PRINT
      :
1990 RETURN
```

Risultato:

In modo immediato pulisce il video e riporta il cursore in posizione di riposo.

In modo differito:

LINEA 100: pulisce il vidèo e riporta il cursore in posizione di riposo.

LINEA 110: esegue una subroutine che effettua delle stampe sul video.

LINEA 120: effettua la pulizia del video, utilizzato dalla routine di stampa, e riporta il cursore in posizione di riposo.

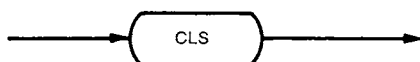
LINEE 1000 ÷ 1990: subroutine che effettua delle stampe su video. Al termine di questa subroutine, il video è riempito di scritte varie, che verranno cancellate alla linea 120.

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

CLS (Comando-istruzione)

Cancella il video, riempiendo tutte le posizioni con spazi, e colorandolo col colore di Background.

Per OLIVETTI M20 si presta a cancellare il contenuto di una determinata finestra.

Formalismo sintattico:

Esempio:

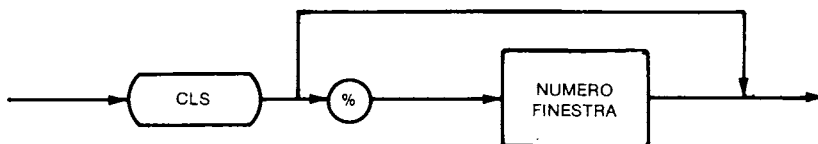
```
10 CLS
20 PRINT "MENU"
30 :
```

Risultato:

LA LINEA 10 cancella il video.
LA LINEA 20 stampa il messaggio specificato.

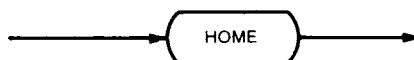
Note:

— CLS riporta anche il cursore nella posizione di riposo.

Variazione per: Olivetti M20.

HOME (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Riempie tutte le posizioni del video con spazi e riporta il cursore in posizione di riposo (in alto a sinistra sullo schermo).

Formalismo sintattico:

Esempio:

```
100 GR
110 COLOR = 5
120 HLIN 0, 10 AT 15
130 HOME
140 COLOR = 7
    :
200 END
```

Risultato:

LA LINEA 100 cancella il video e lo predispone in modo grafico.
LA LINEA 110 definisce il colore che deve essere usato.
LA LINEA 120 traccia una linea orizzontale da colonna 0 a colonna 10 sulla riga 15.
LA LINEA 130 cancella il video e riporta il cursore alla posizione di riposo, in alto a sinistra.
LA LINEA 140 definisce un nuovo colore da utilizzare.

Note:
— Lo stesso effetto si può ottenere con "CALL —936" oppure, in modo immediato, premendo il tasto "ESC", quindi il tasto "@" e il tasto "RETURN".

CAPITOLO 5

CORREZIONI AD UN PROGRAMMA

GENERALITA'

Dopo che un programma è stato scritto, può sorgere la necessità di doverlo modificare, o perchè contiene errori o perchè lo si vuole ampliare o ridurre o migliorare.
Le modalità di modifica di un programma residente in memoria dipendono dal tipo di elaboratore.

TEXAS TI 99/4A

Per inserire una linea: battere un numero di linea compreso tra quelli delle due istruzioni tra le quali si vuole inserire la nuova linea, digitare l'istruzione voluta, indi battere il tasto "ENTER".
Per cancellare una linea: Battere il numero della linea che si vuole cancellare e subito dopo battere il tasto "ENTER".
Per cancellare gruppi di linee adiacenti: Non è possibile cancellare gruppi di linee con un solo intervento ma è necessario cancellare una linea per volta.
Per modificare una linea: Usare il comando EDIT per visualizzare la linea da modificare, indi variarla in modo opportuno. Dopo la modifica, battere il tasto "ENTER".
È possibile anche riscrivere l'intera istruzione compreso il numero di linea.

COMMODORE VIC 20 E CBM 64

Per inserire una linea: Battere un numero di linea compreso tra quelli delle due istruzioni tra le quali si vuole inserire la nuova linea, digitare l'istruzione voluta indi battere il tasto "ENTER".
Per cancellare una linea: Battere il numero della linea che si vuole cancellare e subito dopo battere il tasto "ENTER".
Per cancellare gruppi di linee adiacenti: Non è possibile cancellare gruppi di linee con un solo intervento, ma è necessario cancellare una linea alla volta.
Per modificare una linea: Effettuare un comando LIST per visualizzare la linea (o le linee) da modificare, posizionarsi col cursore sulla linea da modificare, effettuare le dovute correzioni e battere quindi il tasto "ENTER".
Molta attenzione deve essere posta quando si modifica una linea in cui è stato utilizzato il "modo virgolette".
Tale metodo può anche essere utilizzato per duplicare delle istruzioni, se si modifica anche il numero di linea.
È anche possibile riscrivere l'intera linea, compreso il numero di linea.

SINCLAIR ZX 81 E ZX SPECTRUM

Per inserire una linea: Battere un numero di linea compreso tra quelli delle due istruzioni tra le quali si vuole inserire la nuova linea, digitare l'istruzione voluta, indi battere il tasto "NEW LINE" (ENTER per lo ZX spectrum).

Per cancellare una linea: Battere il numero della linea che si vuole cancellare e subito dopo battere il tasto "NEW LINE" (ENTER per lo ZX spectrum).

Per cancellare gruppi di linee adiacenti: Non è possibile cancellare gruppi di linee con un solo intervento, ma è necessario cancellare una linea alla volta.

Per modificare una linea: Spostare il puntatore di linea, utilizzando i tasti per il movimento verticale (SHIFT e 6 oppure SHIFT e 7), sulla linea da modificare, indi battere il tasto EDIT. La linea si presenta ora nella parte bassa del video e la si può correggere.

È anche possibile riscrivere completamente l'intera linea (compreso il numero di linea).

SHARP MZ 700

Per inserire una linea: Battere un numero di linea compreso tra quelli delle due istruzioni tra le quali si vuole inserire la nuova linea, digitare l'istruzione voluta, indi battere il tasto "CR".

Per cancellare una linea: Battere il numero della linea che si vuole cancellare e subito dopo battere il tasto "CR", oppure usare il comando DELETE.

Per cancellare gruppi di linee adiacenti: Far uso del comando DELETE.

Per modificare una linea: Effettuare un comando LIST per visualizzare la linea (o le linee) da modificare, posizionarsi col cursore sulla linea da modificare, effettuare le dovute correzioni e battere quindi il tasto "CR".

Tale metodo può essere utilizzato per duplicare delle istruzioni, se viene modificato anche il numero di linea.

È anche possibile riscrivere l'intera linea, compreso il numero di linea.

OLIVETTI M 20

Per inserire una linea: Battere un numero di linea compreso tra quelli delle due istruzioni tra le quali si vuole inserire la nuova linea, digitare l'istruzione voluta, indi battere il tasto "CR".

Per cancellare una linea: Battere il numero della linea che si vuole cancellare e subito dopo battere il tasto "CR", oppure usare il comando DELETE.

Per cancellare gruppi di linee adiacenti: Far uso del comando DELETE.

Per modificare una linea: Sostituire l'intera linea, impostando il numero di linea e il suo nuovo contenuto oppure utilizzare l'Editor di linea, richiamato mediante il

comando EDIT. L'editor di linea permette, attraverso comandi specializzati, un certo numero di operazioni anche complesse (cancellazione o inserimenti di un carattere o di una stringa di caratteri, ricerca di un carattere ben determinato, modifica a un carattere, cancellazione della parte di riga a destra del cursore ecc.). Si può uscire dallo stato Editor battendo "CR", e la linea risulterà modificata. Se si usa il comando dell'Editor Q (QUIT), si esce dallo stato Editor ma le modifiche non verranno prese in considerazione.

I.B.M. P.C.

Per inserire una linea: Battere un numero di linea compreso tra quelli delle due istruzioni tra le quali si vuole inserire la nuova linea, digitare l'istruzione voluta, indi battere il tasto di immissione.

Per cancellare una linea: Battere il numero della linea che si vuole cancellare e subito dopo battere il tasto di immissione, oppure usare il comando DELETE.

Per cancellare gruppi di linee adiacenti: Far uso del comando DELETE.

Per modificare una linea: Effettuare un comando LIST per visualizzare la linea (o le linee) da modificare, posizionarsi col cursore sulla linea da modificare, effettuare le dovute correzioni e battere quindi il tasto di immissione.

È possibile riscrivere l'intera linea, compreso il numero di linea.

APPLE II

Per inserire una linea: Battere un numero di linea compreso tra quelli delle due istruzioni tra le quali si vuole inserire la nuova linea, digitare l'istruzione voluta, indi battere il tasto di immissione;

Per cancellare una linea: Battere il numero della linea che si vuole cancellare e subito dopo battere il tasto di immissione, oppure usare il comando DEL.

Per cancellare gruppi di linee adiacenti: Far uso del comando DEL.

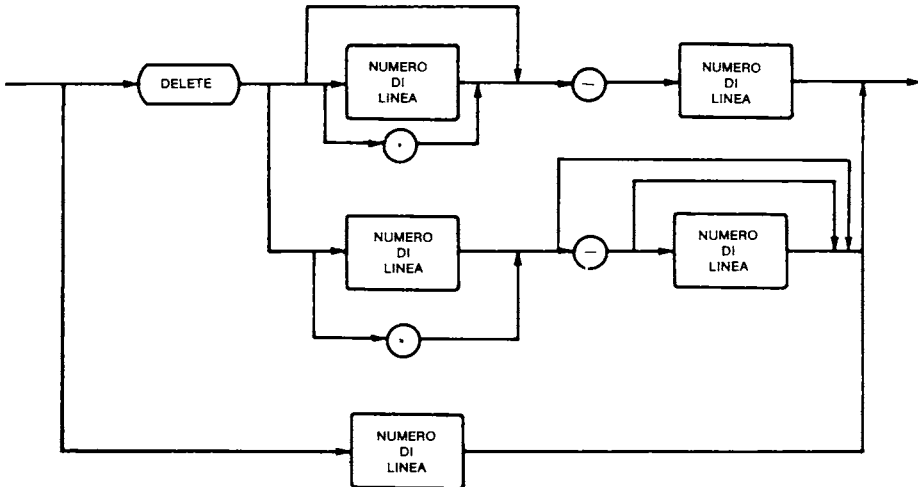
Per modificare una linea: Elencare la riga sullo schermo e usare i comandi di editing per spostare il cursore in modo che esso si disponga direttamente sul primo carattere della riga elencata. Usare quindi i tasti di freccia a destra e REPT per ricopiare i caratteri dallo schermo, modificando quelli che si desidera cambiare. Premere quindi il tasto di immissione. È possibile riscrivere l'intera linea, compreso il numero di linea.

DELETE (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Cancella una linea, o un gruppo di linee, del programma.

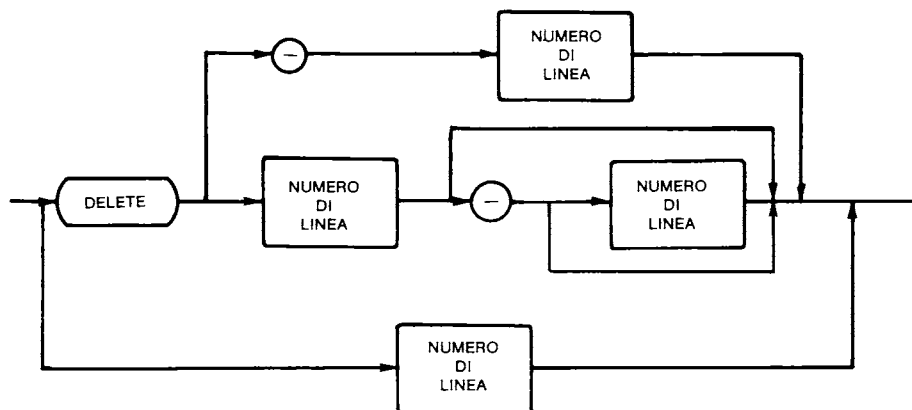
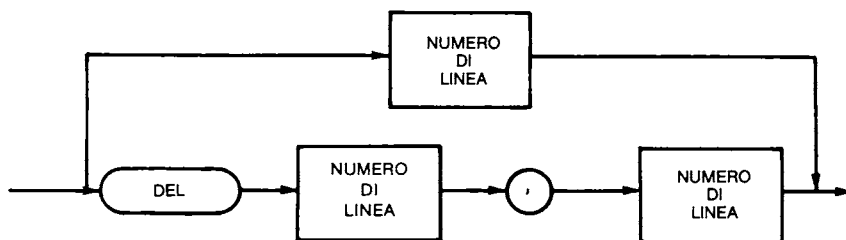
Dopo l'esecuzione del comando DELETE, l'elaboratore ritorna allo stato comandi.

Formalismo sintattico:

Esempi:

1) DELETE 100
2) DELETE 100—200
3) DELETE —100
4) DELETE 100—
5) DELETE .—150
6) 100 [ENTER]

Risultato:

L'esempio 1 cancella la linea 100.
L'esempio 2 cancella tutte le linee dalla 100 alla 200 comprese.
L'esempio 3 cancella tutte le linee dall'inizio del programma fino alla 100 compresa.
L'esempio 4 cancella tutto il programma dalla linea 100 fino alla fine.
L'esempio 5 cancella dalla linea attuale fino alla linea 150 compresa.
L'esempio 6 cancella la linea 100.

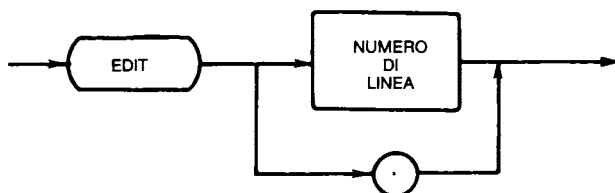
Variazione per: MZ 700

Variazione per: APPLE II

Nota:
— NUMERO DI LINEA è una costante numerica che può assumere valori compresi fra 0 e 65.535.

EDIT (Comando)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Viene utilizzato per entrare nel "Modo Edit", che permette la modifica al programma residente in memoria.

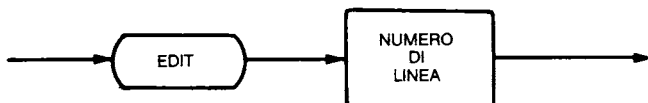
Formalismo sintattico:

Esempio:

```
10 REM PROGRAMMA DI CARICAMENTO
20 REM 20-01-80
   EDIT 20
```

Risultato:

Il comando EDIT fa stampare la linea 20 e consente di modificarla.

Note:
- Dopo che la linea è stata visualizzata, il sistema resta in attesa che venga digitato un comando dell'Editor. Questi comandi formano un linguaggio a sé stante, diverso da elaboratore a elaboratore, per cui rimandiamo ai manuali della macchina per ulteriori spiegazioni.
- Il comando EDIT è valido solo in modo immediato.
- NUMERO DI LINEA è una costante numerica che può assumere valori compresi fra 0 e 65536.

Variazione per: TI 99/4A

CAPITOLO 6

POSIZIONAMENTO CURSORE

GENERALITA'

In taluni elaboratori è possibile modificare la posizione del cursore caratteri (o del cursore grafici), facendolo muovere a piacimento sullo schermo. Tale possibilità è utilizzata, prevalentemente, prima di istruzioni di input/output da video (esempio INPUT, PRINT, ecc.).

Le modalità differiscono a seconda dell'elaboratore.

Per: ZX 81 → vedere opzione AT (con la PRINT)
Per: ZX spectrum → vedere opzione AT (con la PRINT o la INPUT)
Per: MZ 700 → vedere istruzione CURSOR
Per: OLIVETTI M20 → vedere istruzioni CURSOR e CURSOR POINT
Per: I.B.M. P.C. → vedere istruzione LOCATE

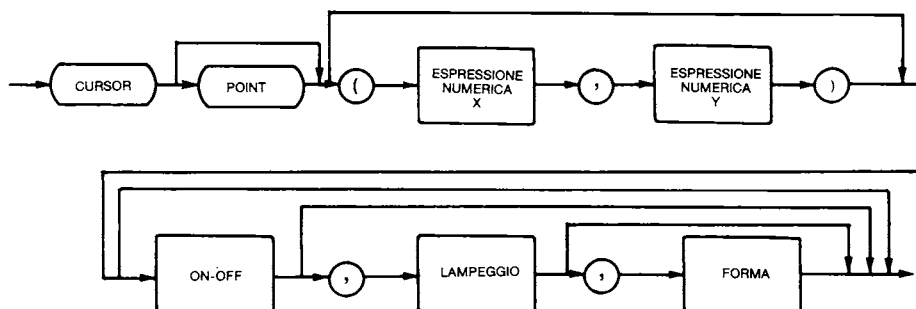
Per: APPLE → vedere istruzioni HTAB e VTAB.

Per: TEXAS TI 99/4A
VIC 20
CBM 64 non esiste la possibilità di posizionare a richiesta il cursore.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

CURSOR (Istruzione)

Specifica la posizione del cursore di testo (o del cursore grafico, solo per OLIVETTI M 20). Su OLIVETTI M 20 specifica anche le caratteristiche dei due cursori.

Formalismo sintattico:

Esempio:

```
10 INPUT "COORDINATA X"; X
20 INPUT "COORDINATA Y"; Y
30 INPUT "CARATTERE DA STAMPARE"; C
40 CLS
50 CURSOR X, Y : PRINT CHR$ (C)
60 FOR I = 1 TO 1000 : NEXT I
70 CURSOR X, Y : PRINT " "
80 GO TO 10
```

Risultato:

Il programma stampa un carattere voluto nella posizione voluta; per far terminare il programma utilizzare il tasto RESET o BREAK o equivalenti.

LINEE 10÷30: chiedono l'introduzione dei valori di riga e di colonna per la posizione del cursore, e il codice ASCII del carattere da visualizzare.

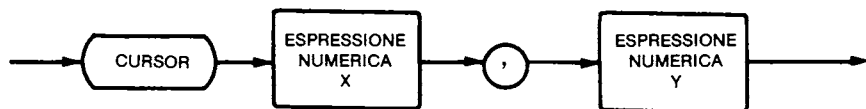
LINEA 40: pulizia del video (per M20).

LINEA 50: posizionamento del cursore nella posizione voluta e stampa del carattere corrispondente al codice ASCII introdotto.

LINEA 60: ciclo di ritardo per permettere la visualizzazione del carattere stampato.

LINEA 70: riposizionamento del cursore e stampa di uno spazio per cancellare il carattere stampato.

LINEA 80: riporta il controllo dell'esecuzione alla linea 10, per rieseguire di nuovo il programma dall'inizio.

Variazione per: MZ700

Note:

— ON-OFF è una espressione numerica che può assumere i seguenti valori:
  0 = il cursore non viene visualizzato;
  1 = il cursore viene visualizzato.

— LAMPEGGIO è una espressione numerica che può assumere i seguenti valori:
  0 = il cursore non viene fatto lampeggiare;
  1÷20 = il cursore viene fatto lampeggiare, con frequenza diversa di lampeggio.

— FORMA è una matrice intera unidimensionale a sei elementi definita dall'utente, che permette di variare la forma del cursore.

— l'opzione POINT viene usata quando si voglia operare sul cursore grafico.

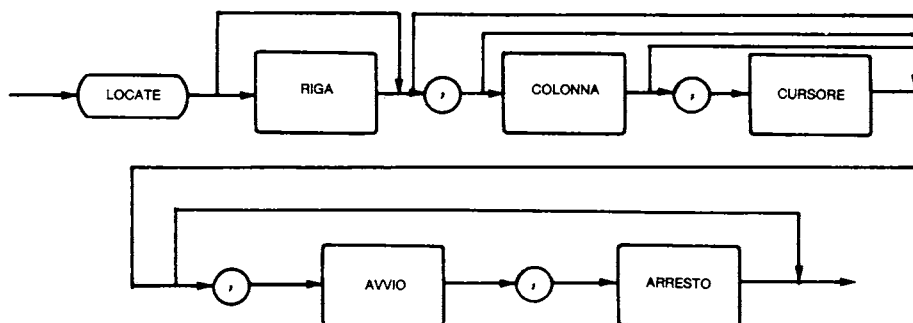
— le espressioni numeriche X e Y assumono valori diversi se si sta agendo sul

cursore di testo rispetto a quando si opera sul cursore grafico. Nel primo caso corrispondono, rispettivamente, alla riga e alla colonna volute, nel secondo caso rappresentano le coordinate di un punto luminoso (pixel) sul video.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

LOCATE (Istruzione)

Posiziona il cursore sullo schermo attivo. I parametri facoltativi attivano e disattivano il lampeggiamento del cursore e definiscono l'ampiezza del cursore lampeggiante.

Formalismo sintattico:

Esempi:

1) LOCATE 1,1
2) LOCATE 10, 15
3) LOCATE 5, 7, 1, 7

Risultati:

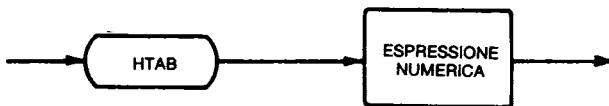
L'esempio 1 posiziona il cursore sulla prima riga, prima colonna.
L'esempio 2 posiziona il cursore sulla riga 10 colonna 15.
L'esempio 3 posiziona il cursore alla riga 5 e colonna 7, rende visibile il cursore e lo visualizza sotto il carattere (avvio e arresto sulla riga di scansione 7).

Note:
— RIGA è un'espressione numerica che può assumere valori compresi fra 1 e 25.
— COLONNA è un'espressione numerica che può assumere valori compresi fra 1 e 40 (oppure fra 1 e 80, in dipendenza dell'ampiezza dello schermo).
— CURSORE è un'espressione numerica che può assumere i seguenti valori:
0 = il cursore non viene visualizzato;
1 = il cursore viene visualizzato.
— AVVIO e ARRESTO sono espressioni numeriche che possono assumere valori compresi fra 1 e 31. Essi individuano, rispettivamente, la riga di scansione iniziale e finale del cursore, e consentono quindi di modificare a piacimento l'ampiezza del cursore.
— CURSORE, AVVIO e ARRESTO non sono utilizzabili in modo grafico.

HTAB (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Sposta il cursore dal bordo di sinistra della riga corrente dello schermo alla posizione data dal valore dell'espressione numerica. Se tale valore è maggiore di 40 (o di suoi multipli) il cursore passerà alle righe successive.

Formalismo sintattico:

Esempi:

1) HTAB 10: PRINT "*"
2) HTAB 21: PRINT "PIPPO"
3) A = 7: HTAB (A): PRINT "*"

Risultati:

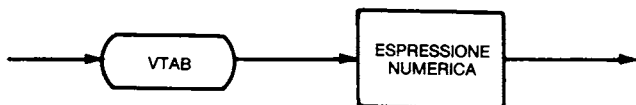
Nell'esempio 1 viene stampato un asterisco alla colonna 10 della riga corrente.
Nell'esempio 2 viene stampata la parola PIPPO dalla colonna 21 della riga corrente.
Nell'esempio 3 viene stampato un asterisco alla colonna 7 della riga corrente.

Note:
— Le colonne sono numerate da 1 a 40 da sinistra verso destra.
— La funzione complementare per le righe è VTAB.
— Gli spostamenti sono sempre relativi al bordo di sinistra della riga corrente.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

VTAB (Istruzione)

Sposta il cursore alla riga data dal valore dell'espressione numerica.

Formalismo sintattico:

Esempi:

1) VTAB 10: PRINT "*"
2) VTAB 21: PRINT "PIPPO"
3) A = 7: VTAB (A): PRINT "*"

Risultati:

Nell'esempio 1 viene stampato un asterisco alla riga 10.
Nell'esempio 2 viene stampato "PIPPO" alla riga 21.
Nell'esempio 3 viene stampato un asterisco alla riga 7.

Note:
— Le righe sono numerate da 1 a 24 dall'alto al basso.
— La funzione complementare per le colonne è HTAB.
— Lo spostamento effettuato dall'istruzione VTAB può essere verso l'alto o verso il basso, ma non sarà mai verso destra o verso sinistra.

CAPITOLO 7

FUNZIONI (OPERATORI FUNZIONALI)

Generalità

Funzioni numeriche

ABS
ACS
ASC
CODE
ASN
ATN
CDBL
CINT
CSNG
COS
CVI, CVS, CVD
EXP
FIX
FRE
INSTR
POS (per stringhe)
INT
LEN
LOG
LN
PEEK
PAI
POS (per il cursore)
RAD
SGN
SIN
SPC
SQR
TAB
TAN
VAL

Funzioni di stringa

CHR$
HEX$
OCT$
LEFT$
MID$
RIGHT$
SEG$
STRING$
TO
STR$
MKI$, MKS$, MKD$
SPACE$
VAL$

GENERALITA'

Un tipo di operatori, diversi da quelli finora presi in considerazione, è dato dagli operatori funzionali, normalmente denominati "funzioni". Essi, applicati ad argomenti numerici o alfanumerici, ritornano un argomento (che a sua volta può essere numero o meno) che è una funzione dell'argomento di partenza (esempio: il seno, la radice quadrata, il logaritmo ecc.).

Tali operatori non sono istruzioni BASIC perchè non possono esistere da soli, ma solo in concomitanza con altre istruzioni BASIC (esempio la PRINT, la LET, ecc.). Un esempio tipico di utilizzo è il seguente:

$\upsilon = F\,(X)$

dove:
υ è il nome di una variabile
F è la funzione
X è un'espressione che forma l'argomento della funzione

Esistono due classi di operatori funzionali:

- Funzioni numeriche: possono avere argomento numerico o alfanumerico, ma ritornano sempre un risultato numerico.
- Funzioni di stringa: possono avere argomenti numerici o alfanumerici, ma ritornano sempre un risultato alfanumerico (stringa).

ABS (Funzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna il valore assoluto dell'espressione numerica considerata.

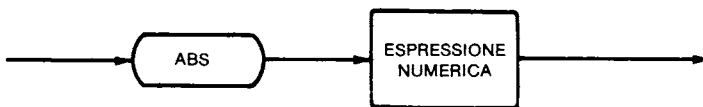
Formalismo sintattico:

Esempio:

```
10 A = 25
20 B = - 52
30 PRINT ABS (A), ABS (B)
40 PRINT ABS (A + B), ABS (A - B)
```

Risultato:

LINEE 10÷20: assegnano i valori specificati alle variabili A e B.
LINEA 30: stamperà a video i seguenti valori: 25 52.
LINEA 40: valuta dapprima il valore delle espressioni numeriche, e poi ne determina il valore assoluto, per cui si avrà la stampa dei seguenti valori: 27 77.

Variante per: ZX81 - ZX SPECTRUM

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

ACS (Funzione)

Ritorna il valore dell'arcocoseno dell'espressione numerica considerata (in radianti).

Formalismo sintattico:

Esempio:

```
10 PRINT "BATTERE IL VALORE DEL COSENO (DA - 1 A 1)"
20 INPUT C
30 A = ACS C
40 G = INT (A * 57.29578)
50 PRINT "L'ANGOLO IL CUI COSENO È"; C; "È DI";
60 PRINT A; "RADIANTI, CORRISPONDENTI A"; G; "GRADI"
70 END
```

Risultato:

LINEA 10: stampa il messaggio specificato.
LINEA 20: attende l'introduzione di un valore numerico compreso tra −1 e 1, che assegna alla variabile C. Supponiamo di digitare 0.
LINEA 30: determina il valore dell'arco (in radianti) il cui coseno è dato dal contenuto della variabile C.
LINEA 40: trasforma i radianti in gradi.
LINEE 50÷60: ottengono, per il valore digitato del coseno (0), la seguente stampa:

L'ANGOLO IL CUI COSENO È 0 È DI 1.5708
RADIANTI, CORRISPONDENTI A 90 GRADI.

LINEA 70: fa terminare l'esecuzione del programma.

Note:

— Tale funzione può essere simulata, se X è il valore in radianti dell'espressione numerica, con l'utilizzo della seguente formula:

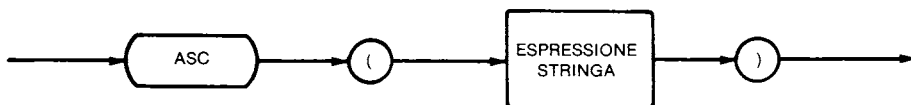
2 * (ATN (1) — ATN (X/ (1 + SQR (1 — X * X))))

— L'espressione numerica dovrà assumere valori compresi fra −1 e +1.
— Volendo il risultato in gradi invece che in radianti, è necessario moltiplicare il risultato per 57.29578.

ASC (Funzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna il codice ASCII del primo carattere della stringa considerata.

Formalismo sintattico:

Esempio:

```
10 A = ASC ("PIPPO")
20 PRINT A
30 PRINT ASC ("ABCDEF")
```

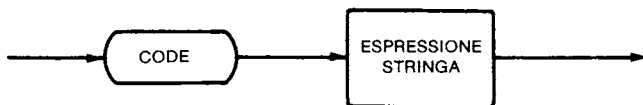
Risultato:

LINEA 10: assegna il codice ASCII del primo carattere della stringa considerata (la P, il cui codice ASCII è 80) alla variabile A.
LINEA 20: stampa il contenuto della variabile A, e cioè: 80
LINEA 30: stampa il codice ASCII del primo carattere della stringa considerata (la A), e quindi: 65.

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CODE (Funzione)

Ritorna il codice ASCII del primo carattere della stringa considerata.

Formalismo sintattico:

Esempio:

```
10 A = CODE "PIPPO"
20 PRINT A
30 PRINT CODE "ABCDEF"
```

Risultato:

LINEA 10: assegna il codice ASCII del primo carattere della stringa considerata (la P, il cui codice ASCII è 80) alla variabile A.
LINEA 20: stampa il contenuto della variabile A, e cioè: 80.
LINEA 30: stampa il codice ASCII del primo carattere della stringa considerata (la A), e quindi: 65.

ASN (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Ritorna il valore dell'arcoseno dell'espressione numerica considerata (in radianti).

Formalismo sintattico:

Esempio:

```
10 PRINT "BATTERE IL VALORE DEL SENO (DA – 1 A 1)"
20 INPUT S
30 A = ASN S
40 G = INT (A * 57.29578)
50 PRINT "L'ANGOLO IL CUI SENO È"; S; "È DI"; A;
60 PRINT "RADIANTI, CORRISPONDENTI A"; G; "GRADI"
70 END
```

Risultato:

LINEA 10: stampa il messaggio specificato.
LINEA 20: attende l'introduzione di un valore numerico compreso tra −1 e 1, che assegna alla variabile S. Supponiamo di digitare 0.5.
LINEA 30: determina il valore dell'arco (in radianti) il cui seno è dato dal contenuto della variabile S.
LINEA 40: trasforma i radianti in gradi.
LINEE 50÷60: ottengono, per il valore digitato del coseno (0.5), la seguente stampa:
L'ANGOLO IL CUI SENO È 0.5 È DI 0.52360
RADIANTI, CORRISPONDENTI A 30 GRADI
LINEA 70: fa terminare l'esecuzione del programma.

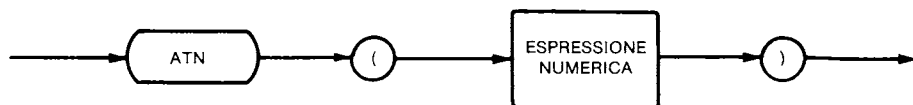
Note:

— Tale funzione può essere simulata, se X è il valore in radianti dell'espressione numerica, con l'utilizzo della seguente formula:
2 * ATN (X/ (1 + SQR (1 − X*X)))
— L'espressione numerica dovrà assumere valori compresi fra −1 e +1.
— Volendo il risultato in gradi invece che in radianti, è necessario moltiplicare il risultato per 57.29578.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

ATN (Funzione)

Ritorna l'arcotangente del valore dell'espressione numerica considerata.

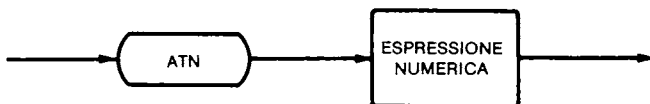
Formalismo sintattico:

Esempi:

1) A = 3: PRINT ATN (A)
2) A = 5/7: PRINT ATN (A)

Risultati:
Nell'esempio 1 verrà stampato: 1.249046
Nell'esempio 2 verrà stampato: .6202495

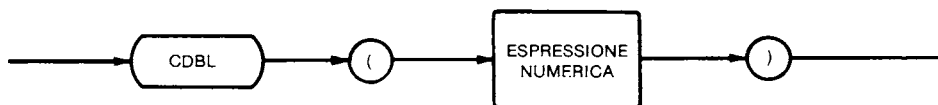
Note:
— Mediante la funzione ATN è possibile calcolare le altre funzioni circolari inverse:

Arcocotangente → ATN (1/X)
Arcosecante → ATN (1/SQR (X * X - 1))
Arcocosecante → ATN (SQR (X * X - 1))
Per l'arcocoseno vedere ACS e per l'arcoseno vedere ASN
— Volendo il risultato in gradi invece che in radianti, si dovrà moltiplicare il risultato parziale per 57.29578.

Variazione per ZX81 - ZX SPECTRUM

CDBL (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Converte il valore dell'espressione numerica considerata in un numero a precisione doppia.

Formalismo sintattico:

Esempio:

```
10 A = 4 : C = 3 : R = A/C
20 PRINT R : PRINT CDBL (R)
```

Risultato:

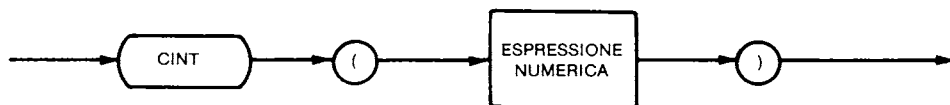
LA LINEA 10 assegna ad A il valore 4 e a B il valore 3; calcola R.
LA LINEA 20 stampa: 1.33333 1.33333373069763.

Note:
— Funzioni equivalenti sono CINT (per interi) e CSNG (per precisione singola).
— Vedere il capitolo 1 per maggiori informazioni riguardo il tipo di variabili.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

CINT (Funzione)

Converte il valore dell'espressione numerica considerata in un numero intero.

Formalismo sintattico:

Esempio:

```
100 A = 12.417
110 PRINT CINT (A), A
```

Risultato:

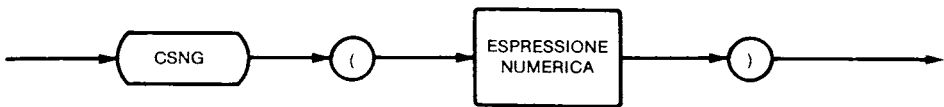
LA LINEA 100 assegna la costante 12.417 alla variabile A.
LA LINEA 110 stampa: 12   12.417

Note:
— Funzioni equivalenti sono CDBL (per doppia precisione) e CSNG (per singola precisione).
— Vedere il capitolo 1 per maggiori informazioni riguardo il tipo di variabili.

CSNG (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Converte il valore dell'espressione numerica considerata in un numero a precisione singola.

Formalismo sintattico:

Esempio:

```
10 A = 4 : C = 3 : R = A/C
20 PRINT CSNG (R)
```

Risultato:

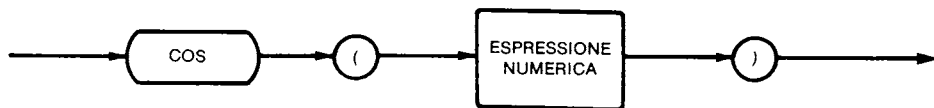
LA LINEA 10 assegna il valore 4 alla variabile A, il valore 3 alla variabile C e calcola R.
LA LINEA 20 stampa: 1.333333.

Note:
— Funzioni equivalenti sono CDBL (per doppia precisione) e CINT (per interi).
— Vedere il capitolo 1 per maggiori informazioni riguardo il tipo di variabili.

COS (Funzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

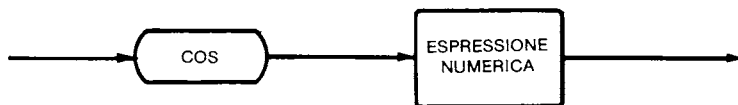
Ritorna il coseno dell'espressione numerica considerata. Il valore dell'espressione numerica deve considerarsi in radianti.

Formalismo sintattico:

Esempi:

1) A = 1: PRINT COS (A)
2) A = 2: PRINT COS (A)

Risultati:

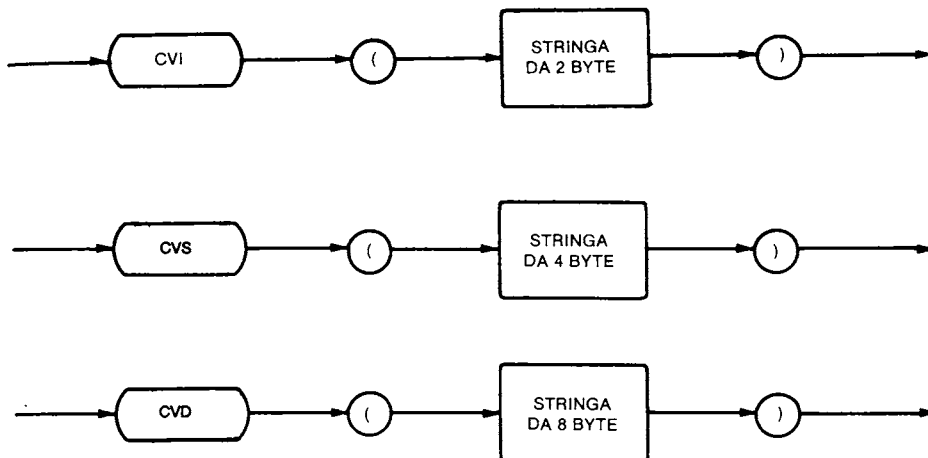
Nell'esempio 1 verrà stampato: .5403023
Nell'esempio 2 verrà stampato: -.4161468

Variazione per ZX81 - ZX SPECTRUM

Note:
— Volendo convertire i radianti in gradi, bisogna moltiplicare per 57.29578.
— Volendo convertire i gradi in radianti, occorre moltiplicarli per 0.0174533.

CVI, CVS, CVD (Funzioni)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Convertono le variabili stringa in numeriche (CVI in intero, CVS in reale a singola precisione, CVD in reale a doppia precisione).

Formalismo sintattico:

Esempio:

```
100 FIELD # 1, 35 AS A$, 2 AS B$, 8 AS C$, 4 AS D$
110 GET # 1
120 A% = CVI (B$)
130 B = CVS (D$)
140 T # = CVD (C$)
```

Risultato:

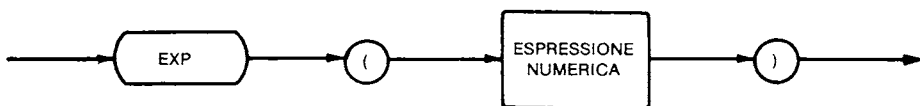
LINEA 100: prepara lo spazio per le variabili di un record nel file ad accesso casuale.
LINEA 110: legge un record del file ad accesso casuale.
LINEA 120: converte il valore letto in B$ in un intero e lo memorizza nella variabile intera A%.
LINEA 130: converte il valore letto in D$ in un numero a precisione singola e lo memorizza nella variabile a precisione singola B.
LINEA 140: converte il valore letto in C$ in un numero a doppia precisione e lo memorizza nella variabile a doppia precisione T#.

Note:
— Le conversioni opposte, cioè da valori numerici a valori stringa, vengono effettuate dalle funzioni MKI$, MKS$ e MKD$.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

EXP (Funzione)

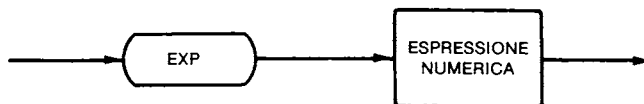
Ritorna l'esponenziale (e^x) dell'espressione numerica considerata.

Formalismo sintattico:

Esempi:

1) PRINT EXP (2)
2) A = 2: PRINT EXP (A)

Risultati:

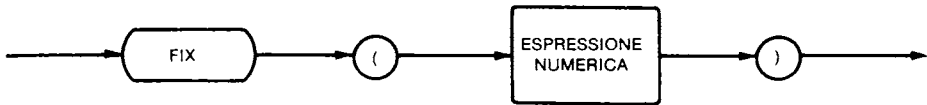
Nell'esempio 1 verrà stampato: 7.389056
Nell'esempio 2 verrà stampato: 7.389056

Variazione per ZX 81 - ZX SPECTRUM

Note:
— È l'opposto della funzione LOG.
— Il valore dell'espressione numerica non deve superare 88.02969.

FIX (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Tronca il valore dell'espressione numerica considerata a un intero.

Formalismo sintattico:

Esempio:

1) PRINT FIX (12.123)
2) A = —21.471: PRINT FIX (A)
3) PRINT FIX (—13.88)

Risultato:

Nell'esempio 1 verrà stampato: 12.
Nell'esempio 2 verrà stampato: −21.
Nell'esempio 3 verrà stampato: −13.

Note:
— Anche le funzioni INT e CINT ritornano valori interi.
— La differenza fra INT e FIX è che INT ritorna il successivo valore più piccolo quando l'espressione numerica è negativa. Per esempio:
INT (− 13.88) ritorna −14

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

FRE (Funzione)

Ritorna il numero di byte di memoria che non sono utilizzati dal BASIC.

Formalismo sintattico:

Esempio:

In modo immediato:
> PRINT FRE (0)

in modo differito:
500 IF FRE (0) < 4096 THEN GOSUB 10000

.

10000 REM * SUBROUTINE MINIMA *

Risultato:

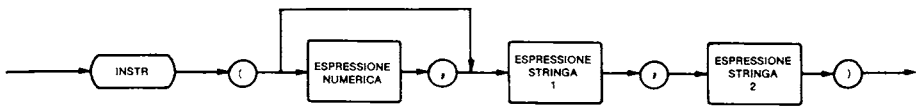
In modo immediato, stampa la quantità (in byte) di memoria a disposizione.
In modo differito, se la quantità di memoria disponibile è inferire a 4 Kbyte, viene eseguita la routine che inizia alla linea 10.000.

Note:
- L'espressione (che può essere numerica o stringa) che accompagna la funzione FRE è un argomento fittizio, richiesto solo per motivi sintattici.
- in alcuni elaboratori (es. I.B.M. P.C.), la funzione FRE con valori stringa causa un riordino dei dati utili prima di dichiarare lo spazio di memoria libero.

INSTR (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Ricerca il primo ripetersi della stringa 2 nella stringa 1 e ritorna la posizione in cui si trova la corrispondenza. La posizione relativa, data dal valore dell'espressione numerica, definisce la posizione di inizio per la ricerca nella stringa 1.

Formalismo sintattico:

Esempi:

1) PRINT INSTR (1, "PIPPO", "I")
2) PRINT INSTR (1, "PIPPO", "P")
3) PRINT INSTR (2, "PIPPO", "P")
4) PRINT INSTR ("PIPPO", "T")

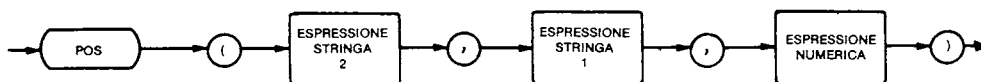
Risultati:
Nell'esempio 1 verrà stampato: 2
Nell'esempio 2 verrà stampato: 1
Nell'esempio 3 verrà stampato: 3
Nell'esempio 4 verrà stampato: 0

Note:
— Il valore dell'espressione numerica non deve essere superiore alla lunghezza della stringa 1 e, in generale, non deve superare 255, che è la massima lunghezza consentita alle stringhe.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

POS (Funzione)

Ricerca il primo ripetersi della stringa 1 nella stringa 2 e ritorna la posizione in cui si trova la corrispondenza. La posizione relativa, data dal valore dell'espressione numerica, definisce la posizione di inizio per la ricerca nella stringa 2.

Formalismo sintattico:

Esempi:

1) PRINT POS ("PIPPO", "I", 1)
2) PRINT POS ("PIPPO", "P", 1)
3) PRINT POS ("PIPPO", "P", 2)
4) PRINT POS ("PIPPO", "T", 1)

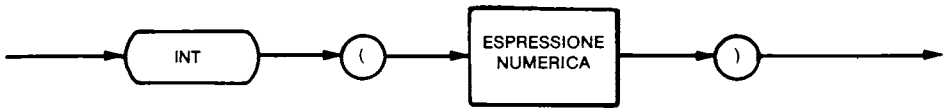
Risultati:
Nell'esempio 1 verrà stampato: 2
Nell'esempio 2 verrà stampato: 1
Nell'esempio 3 verrà stampato: 3
Nell'esempio 4 verrà stampato: 0

Note:
— Il valore dell'espressione numerica non deve essere superiore alla lunghezza della stringa 2 e, in generale, non deve superare 255, che è la massima lunghezza consentita alle stringhe.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

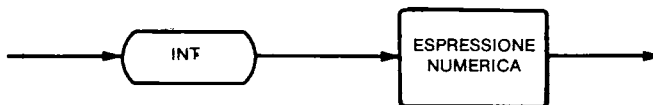
INT (Funzione)

Ritorna l'intero più grande che è minore o uguale al valore dell'espressione numerica considerata.

Formalismo sintattico:

Esempi:

1) PRINT INT (3.123)
2) PRINT INT (7.885)
3) PRINT INT (5.611)
4) A = 2 : B = 21.3 : PRINT INT (A + B)
5) PRINT INT (—13.88)

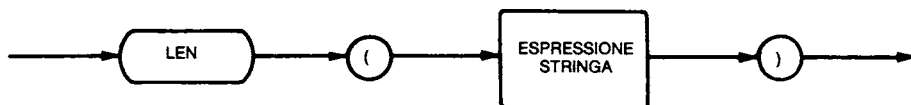
Risultati:
Nell'esempio 1 verrà stampato: 3
Nell'esempio 2 verrà stampato: 7
Nell'esempio 3 verrà stampato: 5
Nell'esempio 4 verrà stampato: 23
Nell'esempio 5 verrà stampato: −14

Variazione per: ZX81 - ZX SPECTRUM

Note:
— Anche le funzioni FIX e CINT ritornano valori interi.
— La differenza fra FIX e INT è che FIX non ritorna il successivo valore più piccolo quando l'espressione numerica è negativa. Per esempio:
FIX (−13.88) ritorna −13

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

LEN (Funzione)

Ritorna il numero di caratteri (cioè la lunghezza effettiva) della stringa considerata.

Formalismo sintattico:

Esempi:

1) PRINT LEN ("PIPPO")
2) A$ = "120684": PRINT LEN (A$)
3) INPUT B$: PRINT LEN (B$)

Risultati:
Nell'esempio 1 verrà stampato: 5.
Nell'esempio 2 verrà stampato: 6.
Nell'esempio 3 verrà stampato il numero di caratteri della stringa che viene digitata a video. Se ad esempio si digita la parola "TOPOLINO", verrà stampato a video il valore 8.

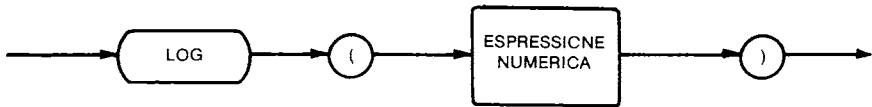
Variazione per: ZX81 - ZX SPECTRUM

Note:
— Il valore ritornato dalla funzione LEN è un intero compreso fra 0 e 255.

LOG (Funzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna il logaritmo naturale (in base e) dell'espressione numerica considerata. Nel caso dello SHARP MZ 700 viene ritornato il logaritmo decimale.

Formalismo sintattico:

Esempio:

Per tutti, tranne SHARP MZ-700
1) PRINT LOG (2)
2) A = 2: PRINT LOG (A)

Per MZ-700:
3) PRINT LOG (2)

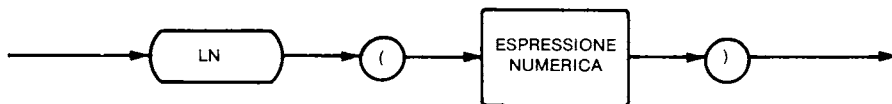
Risultati:
Nell'esempio 1 verrà stampato: .693147
Nell'esempio 2 verrà stampato: .693147
Nell'esempio 3 verrà stampato: .301029

Note:
— il valore dell'espressione numerica deve essere positivo, altrimenti si verificherà un errore
— per convertire un logaritmo decimale in un logaritmo naturale, bisogna moltiplicare il logaritmo decimale per 2.302585
— per convertire un logaritmo naturale in un logaritmo comune (o decimale), bisogna moltiplicare il logaritmo naturale per 0.4342945.

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

LN (Funzione)

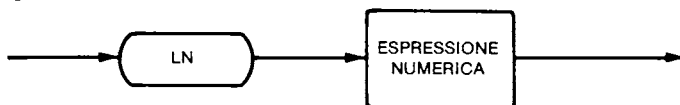
Ritorna il logaritmo naturale (in base e) dell'espressione numerica considerata.

Formalismo sintattico.

Esempio:

```
10 A = 2
20 PRINT LN (A)
30 B = LN (2)
40 PRINT B
```

Risultato:

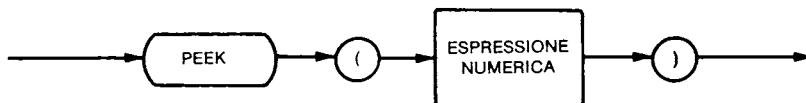
LINEA 10: assegna il valore 2 alla variabile numerica A.
LINEA 20: calcola il logaritmo naturale del contenuto della variabile A e lo evidenzia a video: .693147.
LINEA 30: calcola il logaritmo naturale (in base "e") della costante numerica 2 e lo assegna alla variabile B.
LINEA 40: stampa a video il contenuto di B, e cioè: .693147.

Variazione per ZX81 - ZX SPECTRUM

Note:
— il valore dell'espressione numerica deve essere positivo, altrimenti si verificherà un errore.
— per convertire un logaritmo decimale in un logaritmo naturale, bisogna moltiplicare il logaritmo decimale (o comune) per 2.302585.
— per convertire un logaritmo naturale in un logaritmo comune (o decimale), bisogna moltiplicare il logaritmo naturale per 0.4342945.

PEEK (Funzione)

	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
	OLIVETTI M20
X	APPLE

Ritorna il byte letto dalla posizione di memoria indicata dal valore dell'espressione numerica considerata.
Il valore ritornato sarà un intero compreso tra 0 e 255.

Formalismo sintattico:

Esempio:

```
10 A = PEEK (10000)
20 B = PEEK (10001)
30 PRINT A, B
40 POKE 10000, (PEEK (10000) AND 24) + 1
50 PRINT PEEK (10000)
```

Risultato:

Supponendo che agli indirizzi 10000 e 10001 siano memorizzati i valori 12 e 53 si otterrà:

LA LINEA 10 assegna alla variabile A il contenuto dell'indirizzo di memoria 10000.
LA LINEA 20 assegna alla variabile B il contenuto dell'indirizzo di memoria 10001.
LA LINEA 30 stampa: 12 53.
LA LINEA 40 memorizza all'indirizzo 10000 il valore 9. Infatti: PEEK (10000) = 12; 12 AND 24 = 8; 8 + 1 = 9.
LA LINEA 50 stamperà quindi: 9.

Variazione per: ZX81 - ZX SPECTRUM

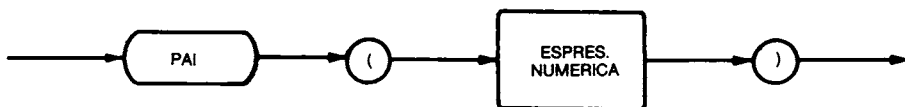
Note:

— POKE è l'istruzione complementare alla funzione PEEK.
— la funzione PEEK e l'istruzione POKE vengono spesso usate, assieme alla funzione USR, per generare ed eseguire sottoprogrammi in linguaggio macchina.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

PAI (Funzione)

Ritorna il valore dell'espressione numerica considerata moltiplicato per il pi-greco (3.141593).

Formalismo sintattico:

Esempio:

```
10 INPUT "RAGGIO ="; R
20 IF R = 0 THEN 80
30 C = PAI (2) * R
40 A = PAI (1) * R ↑ 2
50 V = PAI (4/3) * R ↑ 3
60 PRINT C, A, V
70 GO TO 10
80 END
```

Risultato:

LINEA 10: stampa il messaggio specificato, attende che l'utente introduca un valore da tastiera e lo assegna alla variabile R (Raggio). Supponiamo di introdurre il valore 10.
LINEA 20: se è stato digitato 0, sposta il controllo dell'esecuzione alla linea 80.
LINEA 30: calcola la circonferenza del cerchio di raggio R.
LINEA 40: calcola l'area del cerchio di raggio R.
LINEA 50: calcola il volume della sfera di raggio R.
LINEA 60: stampa i valori calcolati in precedenza. Nel caso R=10 avremo:
62.83186   314.1593   4188.7906.
LINEA 70: sposta il controllo dell'esecuzione alla linea 10 per rieseguire di nuovo il programma.
LINEA 80: fa terminare l'esecuzione del programma.

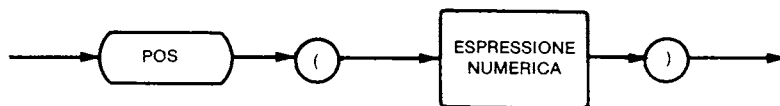
Nota:
— Vedere anche la variabile di sistema PI.

POS (Funzione)

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna la colonna (o la riga, per OLIVETTI M20) in cui si trova attualmente il cursore di testo. Il valore ritornato sarà compreso tra 1 e 40 oppure tra 1 e 80 in base all'impostazione di WIDTH (per I.B.M. P.C.) mentre per OLIVETTI M20 il valore ritornato dipenderà dalle dimensioni della finestra attuale. Per VIC 20 e CBM 64 il valore ritornato è compreso tra 1 e 80.

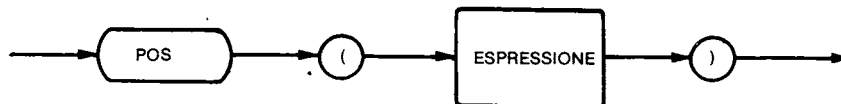
L'espressione numerica viene considerata un argomento fittizio per I.B.M. P.C., VIC 20, CBM 64 e Apple, mentre per OLIVETTI M 20 indica se il valore ritornato si riferisce alla posizione orizzontale o verticale del cursore.

Formalismo sintattico:

Esempio:

```
10 LOCATE 12, 30
20 X = POS (0)
30 PRINT X
```

Risultato:

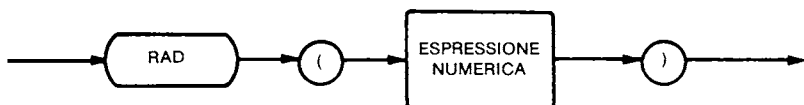
LINEA 10: il cursore viene posizionato alla riga 12 e colonna 30.
LINEA 20: il valore della colonna alla quale si trova il cursore viene assegnato alla variabile numerica X.
LINEA 30: il contenuto della variabile X viene stampato, per cui si avrà: 30.

Variazione per: APPLE II

Note:
- la funzione equivalente è CSRLIN (variabile di sistema), che contiene la coordinata verticale del cursore, cioè il numero di riga (tranne che in M20).
- una funzione simile a POS è LPOS, che ritorna la posizione della testina di stampa.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

RAD (Funzione)

Converte il valore dell'espressione numerica considerata (espresso in gradi) nell'equivalente valore in radianti.

Formalismo sintattico:

Esempio:

```
10 INPUT "GRADI CENTIGRADI"; G
20 R = RAD (G)
30 C = COS (R) : S = SIN (R)
40 PRINT "IL SENO DI"; G; "GRADI È"; S;
50 PRINT "MENTRE IL COSENO È"; C
60 END
```

Risultato:

LINEA 10: attende che venga introdotto da tastiera un valore numerico, che assegna alla variabile G (supponiamo 30).

LINEA 20: converte il valore che è stato introdotto da tastiera in radianti e lo assegna alla variabile R. Nel caso di 30°, avremo R = 0.523599.

LINEA 30: calcola il coseno e il seno dell'angolo espresso in radianti.

LINEE 40÷50: col valore di 30° introdotto, stamperanno a video la seguente frase: IL SENO DI 30 GRADI È 0.5 MENTRE IL COSENO È 0.8660.

LINEA 60: fa terminare l'esecuzione del programma.

Note:

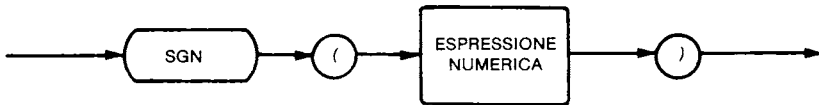
— ove non esista questa funzione, e possibile ottenere il valore in radianti moltiplicando i gradi per 0.0174532.

— per convertire invece gli archi da radianti a gradi, è necessario moltiplicare i radianti per 57.29578.

SGN (Funzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

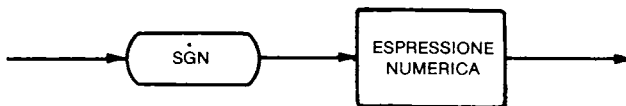
Ritorna il segno dell'espressione numerica considerata. (−1 se l'espressione è negativa, 0 se è zero e 1 se è positiva).

Formalismo sintattico:

Esempi:

1) PRINT SGN (0)
2) PRINT SGN (5)
3) PRINT SGN (—7)

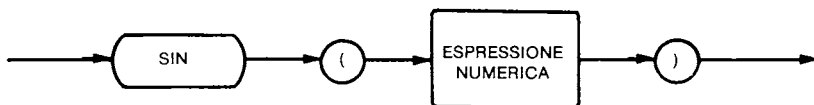
Risultato:

Nell'esempio 1 verrà stampato: 0
Nell'esempio 2 verrà stampato: 1
Nell'esempio 3 verrà stampato: −1

Variazione per: ZX81 - ZX SPECTRUM

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

SIN (Funzione)

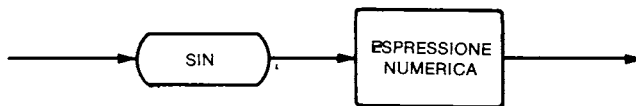
Ritorna la funzione trigonometrica del seno del valore dell'espressione numerica considerata.

Formalismo sintattico:

Esempio:

```
100 FOR I = 1 TO 3
110 PRINT SIN (I)
120 NEXT I
```

Risultato:

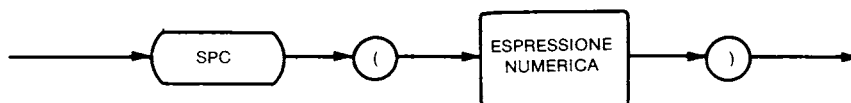
Le linee 100 e 120 innescano un ciclo di tre ripetizioni.
La linea 110 stamperà:
.841470985
– .909297427
– .141120008

Variazione per: ZX81 - ZX SPECTRUM

Note:
— il valore dell'espressione numerica deve essere considerato in radianti
— volendo convertire i radianti in gradi, bisogna moltiplicarli per 57.29578
— volendo convertire i gradi in radianti, occorre moltiplicarli per 0.0174533.

SPC (Funzione)

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

In una istruzione PRINT salta un certo numero di spazi dati dal valore dell'istruzione numerica considerata (vedere anche l'istruzione PRINT).

Formalismo sintattico:

Esempio:

1) PRINT "PIPPO" SPC (3) "PLUTO"
2) PRINT "PIPPO" SPC(A) "PLUTO"

Risultato:

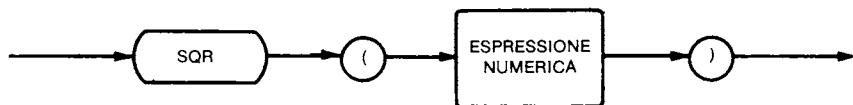
Nell'esempio 1 verrà stampato: PIPPO ... PLUTO
Nell'esempio 2, supponendo che A contenga 2, verrà stampato: PIPPO .. PLUTO.

Note:
— Il valore dell'espressione numerica deve essere compreso tra 0 e 255.
— La funzione SPC può essere utilizzata solo con le istruzioni PRINT, PRINT# e LPRINT (anche con l'opzione USING).

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

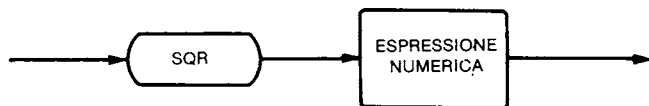
SQR (Funzione)

Ritorna la radice quadrata del valore dell'espressione numerica considerata.

Formalismo sintattico:

Esempi:

1) A = 9 : PRINT SQR (A)
2) A = 0 : PRINT SQR (A)
3) PRINT SQR (2 * 3)

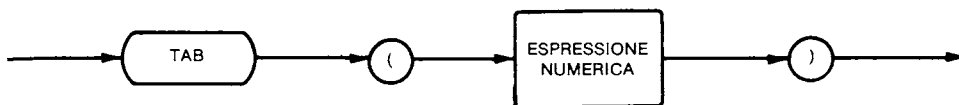
Risultati:
Nell'esempio 1 verrà stampato: 3
Nell'esempio 2 verrà stampato: 0
Nell'esempio 3 verrà stampato: 2.449489

Variazione per: ZX81 - ZX SPECTRUM

Note:
— Il valore dell'espressione numerica deve essere positivo (maggiore o uguale a zero), altrimenti si verificherà un errore.

TAB (Funzione)

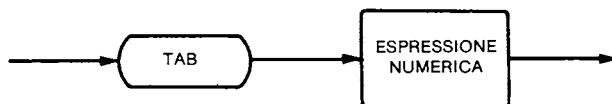
X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

In una istruzione PRINT sposta il cursore alla colonna data dal valore dell'espressione numerica.

Formalismo sintattico:

Esempio:

1) PRINT TAB (2); "PIPPO"
2) A = 3: PRINT TAB (A); "PIPPO"
3) PRINT "PLUTO"; TAB (15); "TOPOLINO"

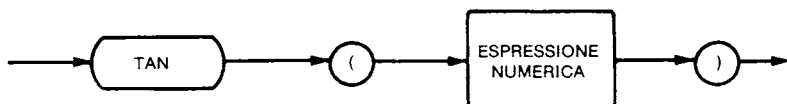
Risultato:
Nell'esempio 1 verrà stampata la parola: PIPPO partendo dalla colonna 2.
Nell'esempio 2 verrà stampata la parola: PIPPO partendo dalla colonna 3.
Nell'esempio 3 verranno stampate le due parole con nove spazi tra loro:
PLUTO .... TOPOLINO.

Variazione per: ZX81 - ZX SPECTRUM

Note:
— il valore dell'espressione numerica deve essere compreso fra 1 e 255.
— se l'attuale posizione del cursore è già superiore al valore dell'espressione numerica, il cursore si sposterà alla colonna voluta, ma sulla riga successiva.
— la funzione TAB può essere utilizzata solo con le istruzioni PRINT, PRINT# e LPRINT (anche con l'opzione USING).

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

TAN (Funzione)

Ritorna la tangente trigonometrica del valore dell'espressione numerica considerata.

Formalismo sintattico:

Esempio:

```
100 FOR I = 1 TO 3
110 PRINT TAN (I)
120 NEXT I
```

Risultato:
Le linee 100 e 120 innescano un ciclo di tre ripetizioni.
La linea 110 stampa:

```
  1.55740772
- .21850987
- .142546543
```

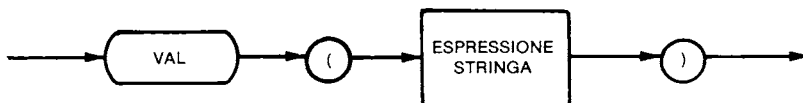
Variazione per: ZX81 - ZX SPECTRUM

Note:
— Il valore dell'espressione numerica deve essere considerato in radianti.
— Volendo convertire i radianti in gradi, bisogna moltiplicarli per 57.29578.
— Volendo convertire i gradi in radianti, occorre moltiplicarli per 0.0174533.

VAL (Funzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna il valore numerico della stringa considerata.

Formalismo sintattico:

Esempi:

1) PRINT VAL ("751.23")
2) A$ = "1980": PRINT VAL (A$)
3) B = VAL (A$): PRINT B

Risultati:
Nell'esempio 1 verrà stampato il valore numerico: 751.23.
Nell'esempio 2 verrà stampato il valore numerico: 1980.
Nell'esempio 3 il valore contenuto nella variabile stringa A$ ("1980"), viene convertito in un valore numerico e memorizzato nella variabile B. Si ottiene poi la stampa: 1980.

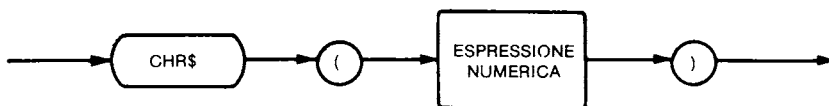
Variazione per: ZX81 - ZX SPECTRUM

Note:
— la funzione VAL elimina eventuali spazi che precedono un valore numerico nella stringa
— se il primo carattere della stringa (escluso lo spazio) non è numerico, verrà ritornato 0 (zero)
— la funzione inversa è STR$.

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

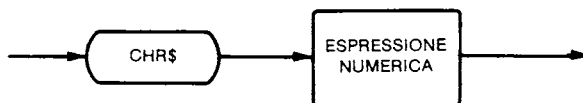
CHR$ (Funzione)

Converte il valore dell'espressione numerica considerata nel carattere corrispondente. Il valore dell'espressione (considerato come codice ASCII) deve essere compreso tra 0 e 255.

Formalismo sintattico:

Esempi:

1) PRINT CHR$ (65)
2) PRINT CHR$ (66)
3) A = 65: PRINT CHR$ (A)

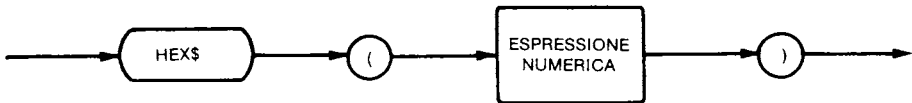
Risultati:
Nell'esempio 1 verrà stampato: A
Nell'esempio 2 verrà stampato: B
Nell'esempio 3 verrà stampato: A

Variazione per: ZX81 - ZX SPECTRUM

Note:
— la funzione inversa è la ASC (oppure CODE)
— l'esecuzione della funzione CHR$ è uno dei modi di utilizzare alcune particolari possibilità, codificate normalmente con codici ASCII inferiori al 32.

HEX$ (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Ritorna una stringa che rappresenta il valore esadecimale dell'espressione numerica considerata.

Formalismo sintattico:

Esempi:

1) PRINT HEX$ (7)
2) A$ = HEX$ (15): PRINT A$

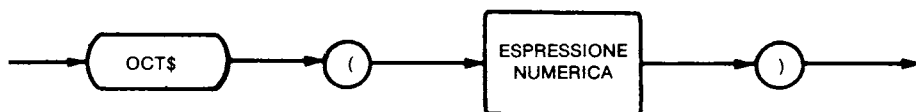
Risultati:
Nell'esempio 1 verrà stampato: 7
Nell'esempio 2 verrà stampato: F

Note:
— l'espressione numerica può assumere valori compresi tra −32768 e 65535
— se il valore dell'espressione numerica è negativo, si effettua un'operazione simile a quella del complemento a due. Cioè:
HEX$ (−X) corrisponde a HEX$ (65.536 −X)
— altre funzioni di conversione sono:
RAD
VAL
OCT$
STR$

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

OCT$ (Funzione)

Ritorna una stringa che rappresenta il valore ottale dell'espressione numerica considerata.

Formalismo sintattico:

Esempi:

1) PRINT OCT$ (8)
2) A$ = OCT$ (37): PRINT A$

Risultati:
Nell'esempio 1 verrà stampato: 10
Nell'esempio 2 verrà stampato: 45

Note:
— l'espressione numerica può assumere valori compresi tra −32768 e 65535
— se il valore dell'espressione numerica è negativo, si effettua un'operazione simile a quella del complemento a due. Cioè.:
OCT$ (−X) corrisponde a: OCT$ (65.536 − X)
— altre funzioni di conversione sono:
RAD
VAL
HEX$
STR$

LEFT$ (Funzione)

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna la parte più a sinistra della stringa considerata. Il numero di caratteri ritornati dipendono dal valore dell'espressione numerica.

Formalismo sintattico:

Esempi:

1) PRINT LEFT$ ("ABCDEFG", 4)
2) A$ = "TOPOLINO" : PRINT LEFT$ (A$, 3)
3) A = 4: A$ = "PLUTO": PRINT LEFT$ (A$, A)

Risultati:
Nell'esempio 1 verrà stampato: ABCD
Nell'esempio 2 verrà stampato: TOP
Nell'esempio 3 verrà stampato: PLUT

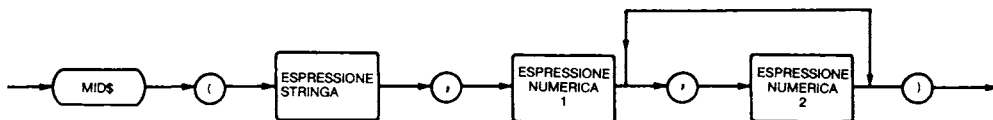
Note:
— l'espressione numerica può assumere valori compresi fra 0 e 255
— se il valore dell'espressione numerica è maggiore del numero di caratteri che compongono la stringa, allora verrà ritornata l'intera stringa
— se il valore dell'espressione numerica è uguale a zero, viene ritornata una stringa nulla
— altre funzioni per manipolare le stringhe sono le seguenti:
MID$
RIGHT$
SEG$
STRING$
TO

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

MID$ (Funzione)

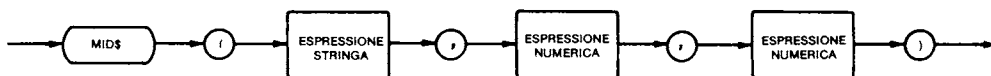
Ritorna la parte richiesta di una determinata stringa.

La posizione iniziale della stringa da estrarre è data dal valore della prima espressione numerica, mentre la lunghezza della porzione di stringa da estrarre è data dal valore della seconda espressione numerica.

Formalismo sintattico:

Esempi:

1) PRINT MID$ ("ABCDEFGHILMN", 6, 3)
2) A$ = "TOPOLINO": PRINT MID$ (A$, 3, 4)
3) A$ = "TOPOLINO": A = 3: B = 1: PRINT MID$ (A$, A, A + B)

Risultati:
Nell'esempio 1 verrà stampato: FGH
Nell'esempio 2 verrà stampato: POLI
Nell'esempio 3 verrà stampato: POLI

Variante per MZ - 700

Note:
- l'espressione numerica 1 può assumere valori compresi fra 1 e 255
- l'espressione numerica 2 può assumere valori compresi fra 0 e 255
- se il valore dell'espressione numerica 1 è maggiore del numero di caratteri della stringa, oppure se il valore dell'espressione numerica 2 è uguale a zero, verrà ritornata una stringa nulla
- in I.B.M. P.C. è possibile utilizzare MID$ anche come istruzione, con un formato: MID$ (X$, n [,m]) = Y$

— altre funzioni per manipolare le stringhe sono le seguenti:
LEFT$
RIGHT$
SEG$
STRING$
TO

RIGHT$ (Funzione)

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna i caratteri più a destra della stringa considerata. Il numero di caratteri che vengono ritornati dipendono dal valore dell'espressione numerica considerata.

Formalismo sintattico:

Esempi:

1) PRINT RIGHT$ ("ABCDEFG", 4)
2) A$ = "TOPOLINO": PRINT RIGHT$ (A$, 4)
3) A = 3: A$ = "PLUTO": PRINT RIGHT$ (A$, A)

Risultati:
Nell'esempio 1 verrà stampato: DEFG
Nell'esempio 2 verrà stampato: LINO
Nell'esempio 3 verrà stampato: UTO

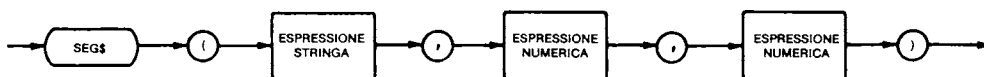
Note:
— l'espressione numerica può assumere valori compresi fra 0 e 255
— se il valore dell'espressione numerica è maggiore del numero di caratteri che compongono la stringa, allora verrà ritornata l'intera stringa
— se il valore dell'espressione numerica è uguale a zero, viene ritornata una stringa nulla
— altre funzioni per manipolare le stringhe sono le seguenti:
LEFT$
MID$
SEG$
STRING$
TO

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

SEG$ (Funzione)

Ritorna una parte della stringa indicata.

La parte di stringa ritornata inizia dal carattere indicato dal valore della prima espressione numerica, ed ha una lunghezza indicata dal valore della seconda espressione numerica.

Formalismo sintattico:

Esempio:

```
100 A$ = "TOPOLINO E PIPPO"
110 T$ = SEG$ (A$, 1, 8)
120 P$ = SEG$ (A$, 12, 5)
130 B$ = SEG$ (T$, 3, 4)
140 PRINT T$
150 PRINT P$
160 PRINT B$
```

Risultato:

LINEA 100: assegna il valore specificato alla variabile stringa A$.

LINEA 110: assegna alla variabile stringa T$ la sottostringa di A$ che inizia dal primo carattere e che è lunga 8 caratteri.

LINEA 120: assegna alla variabile stringa P$ la sottostringa di A$ che inizia dal dodicesimo carattere e che è lunga 5 caratteri.

LINEA 130: assegna alla variabile stringa B$ la sottostringa di T$ (che a sua volta è una sottostringa di A$) che inizia dal terzo carattere e che è lunga 4 caratteri.

LINEE 140÷160: stampano le sottostringhe generate, per cui apparirà a video:

TOPOLINO
PIPPO
POLI

Note:

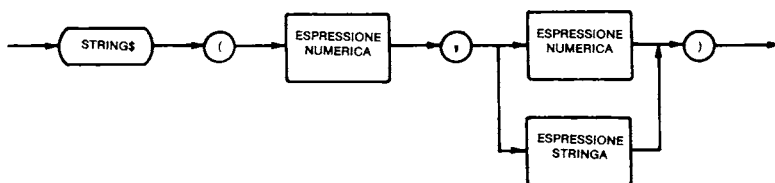
— altre funzioni per manipolare le stringhe sono le seguenti:

LEFT$
MID$
RIGHT$
STRING$
TO

STRINGS$ (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Ritorna una stringa la cui lunghezza è data dal valore della prima espressione numerica e il contenuto è dato da caratteri che hanno il codice ASCII uguale al valore della seconda espressione numerica, oppure dal primo carattere dell'espressione stringa considerata.

Formalismo sintattico:

Esempio:

1) PRINT STRING$ (3, 65)
2) PRINT STRING$ (8, "ABCD")

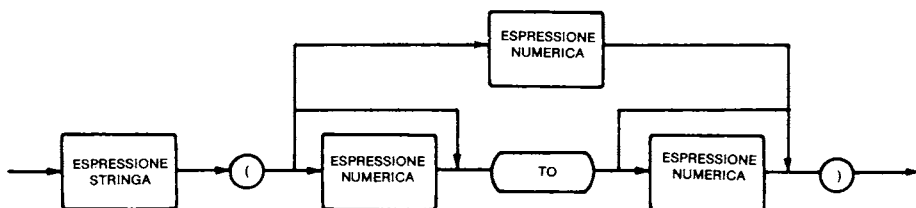
Risultato:
Nell'esempio 1 verrà stampato: AAA
Nell'esempio 2 verrà stampato: AAAAAAAA

Note:
— le espressioni numeriche 1 e 2 possono assumere valori compresi fra 0 e 255
— altre funzioni per manipolare le stringhe sono le seguenti:
LEFT$
MID$
RIGHT$
SEG$
TO

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

TO (Funzione)

Serve per estrarre delle sottostringhe da una stringa data. La sottostringa inizierà dal carattere la cui posizione è data dal valore della prima espressione numerica e terminerà al carattere la cui posizione è data dal valore della seconda espressione numerica.

Formalismo sintattico:

Esempio:

1) PRINT "ABCDEF" (2 TO 5)
2) PRINT "ABCDEF" (TO 5)
3) PRINT "ABCDEF" (2 TO)
4) PRINT "ABCDEF" (TO)
5) PRINT "ABCDEF" (3)

Risultati:
Nell'esempio 1 verrà stampato: BCDE
Nell'esempio 2 verrà stampato: ABCDE
Nell'esempio 3 verrà stampato: BCDEF
Nell'esempio 4 verrà stampato: ABCDEF
Nell'esempio 5 verrà stampato: C

Note:
— la funzione TO può essere anche usata con altre istruzioni. Per esempio si può scrivere:
LET A$ (2 TO 5) = "STRINGA"
— altre funzioni per manipolare le stringhe sono le seguenti:
LEFT$
MID$
RIGHT$
SEG$
STRING$

STR$ (Funzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Ritorna una rappresentazione a stringa del valore dell'espressione numerica considerata.

Formalismo sintattico:

Esempio:

1) PRINT STR$ (123.77)
2) A = 123.77: PRINT STR$ (A)
3) B$ = STR$ (123.77): PRINT B$

Risultato:
Nell'esempio 1 verrà stampato il valore stringa: 123.77
Nell'esempio 2 verrà stampato il valore stringa: 123.77
Nell'esempio 3 il valore numerico 123.77 viene convertito in un valore stringa e memorizzato nella variabile B$. Si ottiene poi la stampa: 123.77

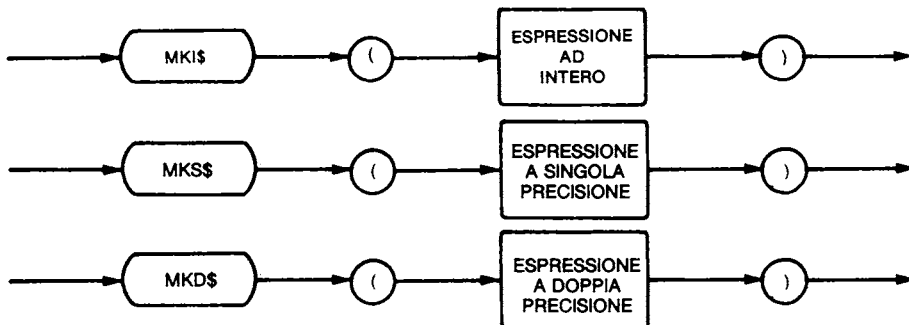
Variazione per ZX81 - ZX SPECTRUM

Note:
— la conversione di un numero in stringa permette di utilizzarlo nelle manipolazioni di stringhe (con LEFT$, MID$, RIGHT$, ecc.)
— la funzione inversa è VAL

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

MKI$, MKS$ MKD$ (Funzioni)

Convertono i valori di tipo numerico in valori di tipo stringa.

Formalismo sintattico:

Esempio:

```
100 INPUT A%
110 INPUT B
120 INPUT T #
130 FIELD # 1, 35 AS A$, 2 AS B$, 8 AS C$, 4 AS D$
140 LSET B$ = MKI$ (A%)
150 LSET C$ = MKD$ (T #)
160 LSET D$ = MKS$ (B)
170 LSET B$ = "COGNOME E NOME"
180 PUT # 1
```

Risultato:

LINEE 100÷120: attendono che vengano digitati valori numerici da tastiera, che assegnano alle variabili specificate.

LINEA 130: prepara lo spazio per le variabili di un record nel file ad accesso casuale.

LINEE 140÷170: preparano le variabili nella memoria di transito del file ad accesso casuale, utilizzando, per quelle di tipo numerico, le funzioni MKI$, MKS$, MKD$.

LINEA 180: scrive il record del file ad accesso casuale.

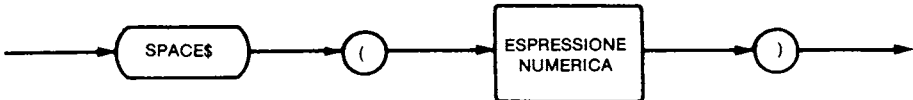
Note:

— Le conversioni opposte, cioè da valori stringa a valori numerici, vengono effettuate dalle funzioni CVI, CVS e CVD.

SPACE$ (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Ritorna una stringa che consiste in un numero di spazi dato dal valore dell'espressione numerica considerata.

Formalismo sintattico:

Esempi:

1) PRINT "PIPPO"; SPACE$ (3); "PLUTO"
2) A = 3: PRINT "PIPPO"; SPACE$ (A); "PLUTO"

Risultati:
Nell'esempio 1 verrà stampato: PIPPO ... PLUTO
Nell'esempio 2 verrà stampato: PIPPO ... PLUTO

Note:
— l'espressione numerica può assumere valori compresi fra 0 e 255
— limitatamente alle istruzioni PRINT, PRINT# e LPRINT (anche con l'opzione USING) è possibile inserire degli spazi anche con le funzioni SPC e TAB.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

VAL$ (Funzione)

Ritorna un valore stringa partendo da una espressione stringa. È una funzione poco utilizzata.

Formalismo sintattico:

Esempio:

```
10 A$ = VAL$ " " "PIPPO" " "
20 PRINT A$
```

Risultato:
Alla linea 10 viene utilizzata la funzione VAL$ per memorizzare una stringa nella variabile stringa A$.
Alla linea 20 viene effettuata la stampa di A$, e si avrà: PIPPO.

CAPITOLO 8

VARIABILI DI SISTEMA

GENERALITA'

Variabili numeriche:

PI

SPEED

ST

SIZE

TI

CSRLIN

Variabili di stringa:

DATE$

TI$

TIME$

GENERALITA'

Sono variabili che il sistema autodefinisce all'atto dell'accensione e che l'utente può utilizzare in qualunque punto del programma BASIC.

Si suddividono anch'esse in numeriche o alfanumeriche.

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

PI (Variabile di sistema)

Contiene il valore 3.141592653589 .... definito come il rapporto tra la circonferenza e il diametro di un cerchio e chiamato π (PI greco).

Tra i molteplici utilizzi di tale variabile di sistema, ricordiamo che è possibile, attraverso π, convertire i gradi in radianti e viceversa. Infatti:

GRADI = RADIANTI * 180/PI

RADIANTI = GRADI * PI/180

Esempio:

```
10 INPUT "RAGGIO ="; R
20 IF R = 0 THEN 80
30 C = 2 * PI * R
40 A = PI * R ↑ 2
50 V = (PI * 4/3) * R ↑ 3
60 PRINT C, A, V
70 GO TO 10
80 END
```

Risultato:

LINEA 10: stampa il messaggio specificato, attende che l'utente introduca un valore da tastiera e lo assegna alla variabile R (raggio). Supponiamo di introdurre il valore 10.

LINEA 20: se è stato digitato 0, sposta il controllo dell'esecuzione alla linea 80.

LINEA 30: calcola la circonferenza del cerchio di raggio R.

LINEA 40: calcola l'area del cerchio di raggio R.

LINEA 50: calcola il volume della sfera di raggio R.

LINEA 60: stampa i valori calcolati in precedenza. Nel caso R=10 avremo:

62.83186   314.1593   4188.7906

LINEA 70: sposta il controllo dell'esecuzione alla linea 10 per rieseguire di nuovo il programma.

LINEA 80: fa terminare l'esecuzione del programma.

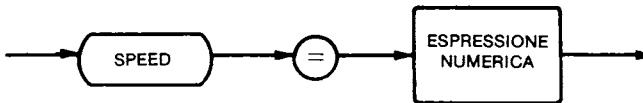
Nota:

— Vedere anche funzione PAI.

SPEED (variabile di sistema)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Definisce la velocità alla quale i caratteri devono essere inviati allo schermo o ad un altro dispositivo di I/O.
La velocità più bassa è 0. La più elevata 255.

Formalismo sintattico:

Esempio:

```
100 INPUT "DAMMI IL VALORE DELLA VELOCITA'"; V
110 SPEED = V
```

Risultato:

LINEA 100: stampa il messaggio specificato e attende che venga introdotto un valore numerico da tastiera, che assegna poi alla variabile V.

LINEA 110: definisce la velocità di stampa dei caratteri uguale al valore della variabile V.

Note:

— il valore dell'espressione numerica deve essere compreso tra 0 e 255.

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

ST (Variabile di sistema)

Contiene il valore corrente dello "status", che dipende dal risultato dell'ultima operazione di I/O.

Esempio:

```
 100 OPEN 1, 8, 4
 110 IF ST > 0 THEN 1000
 120 INPUT # 1, A, B, C$
 130 IF ST = 64 THEN 900
      :
 900 CLOSE 1
 910 END
1000 REM * SUBROUTINE TRATTAMENTO ERRORI *
```

Risultato:

LINEA 100: apre il file sequenziale su disco.

LINEA 110: controlla lo stato dell'operazione di apertura del file. Se ST >0, vuol dire che l'operazione non è andata a buon fine, per cui il controllo viene spostato alla routine di trattamento degli errori.

LINEA 120: legge un record dal file sequenziale.

LINEA 130: controlla il valore dello stato dell'operazione: se è uguale a 64, vuol dire che si è incontrata la fine del file, per cui il controllo viene spostato alla linea 900.

LINEE 900÷910: chiudono il file sequenziale e fanno terminare l'esecuzione del programma.

Note:

— In certi casi al posto di ST viene anche usata la parola STATUS per definire il nome della variabile di sistema. Infatti, in VIC 20 e CBM 64, il nome di una variabile può essere lungo a piacere, ma solo i primi due caratteri sono significativi, per cui ST e STATUS rappresentano un'unica variabile.

SIZE (Variabile di sistema)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Contiene il numero di byte in memoria che possono essere usati per la memorizzazione dei programmi BASIC.

Esempio:

In modo immediato
PRINT SIZE

In modo differito:

```
10 PRINT SIZE
20 DIM A (99) : PRINT SIZE
30 IF SIZE < 1000 THEN END
40 DIM B (249)
```

Risultato:
In modo immediato, stampa il numero di byte a disposizione. Per esempio: 34650.
In modo differito:
LINEA 10: stampa lo spazio ancora a disposizione. Per esempio: 34650.
LINEA 20: dimensiona una matrice di 100 elementi, dopodichè stampa ancora la quantità di byte disponibili. Nel nostro esempio: 34250 (infatti la matrice occupa 400 byte).
LINEA 30: se la memoria disponibile è inferiore a 1000 byte fa terminare l'esecuzione del programma, altrimenti passa ad eseguire la linea successiva.
LINEA 40: dimensiona una matrice di 250 elementi, che occupa quindi 1000 byte di memoria.

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

TI (Variabile di sistema)

Legge l'intervallo dell'orologio. Questo tipo di "clock" è chiamato "jiffy clock". All'accensione del sistema il "jiffy clock" viene impostato a zero. Questo orologio ad intervalli di 1/60 di secondo viene spento durante un I/O su nastro.

Esempio:

```
100 PRINT "BATTI UN TASTO PER INIZIARE"
110 INPUT A$
120 X1 = TI
130 PRINT "BATTI UN TASTO PER FINIRE"
140 INPUT A$
150 X2 = TI
160 PRINT "L'INTERVALLO DI TEMPO È"; 60 * (X2 – X1); "SEC."
```

Risultato:

LA LINEA 100 stampa il messaggio sul video.
LA LINEA 110 attende un input da tastiera.
LA LINEA 120 assegna il valore del timer a X1.
LA LINEA 130 stampa il messaggio sul video.
LA LINEA 140 attende un input da tastiera.
LA LINEA 150 assegna il valore del timer a X2.
LA LINEA 160 stampa l'intervallo di tempo, in secondi, tra il momento dell'inizio e quello della fine.

CSRLIN (Variabile di sistema)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Ritorna la coordinata verticale del cursore (cioè la coordinata di riga). Il valore ritornato sarà compreso tra 1 e 25.

Esempio:

```
10 LOCATE 12, 30
20 Y = CSRLIN
30 PRINT Y
```

Risultato:

LINEA 10: il cursore viene posizionato alla riga 12 e colonna 30.
LINEA 20: il valore della coordinata verticale del cursore, presente nella variabile di sistema CSRLIN, viene assegnato alla variabile numerica Y.
LINEA 30: il contenuto della variabile Y viene stampato, per cui si avrà: 12.

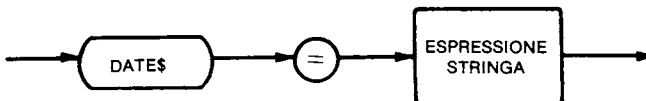
Note:

— la funzione equivalente è POS, che contiene la coordinata orizzontale del cursore, cioè il numero di colonna.

DATE$ (Variabile di sistema)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Contiene il valore della data. Viene ritornata una stringa di 10 caratteri nella forma MM - GG - AAAA, dove MM è il mese, GG il giorno e AAAA è l'anno.

Per impostare inizialmente la data, è necessario scrivere una istruzione col seguente formalismo:

Esempio:

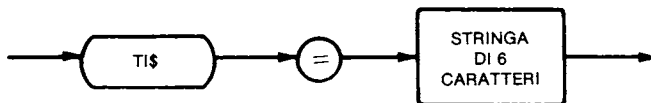
1) A$ = DATE$
2) DATE$ = "08-12-1980"

Risultato:
Nell'esempio 1 viene letta la data dal sistema e viene assegnata alla variabile A$.
Nell'esempio 2 viene assegnato il valore specificato alla data del sistema.

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

TI$ (Variabile di sistema)

Ritorna l'ora mediante una stringa di 6 caratteri nella forma HHMMSS, dove HH è l'ora, MM sono i minuti e SS sono i secondi. All'atto dell'accensione viene impostato a zero. Se lo si vuole impostare diversamente, usare la seguente istruzione:

Formalismo sintattico:

Esempio:

```
  10 REM * INIZIO PROGRAMMA *
  20 TI$ = "000000"
  30 :
     :
8990
9000 REM * FINE PROGRAMMA *
9010 PRINT TI$
9020 END
```

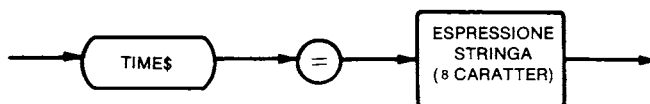
Risultato:

LINEA 10: serve per documentare il programma.
LINEA 20: azzera l'orologio del sistema.
LINEE 30÷8990: contengono il resto del programma.
LINEA 9000: serve per documentare il programma.
LINEA 9010: stampa il valore contenuto nella variabile di sistema TI$. Se ad esempio venisse stampato il seguente valore:
002132
ciò significa che il programma, in totale, è durato 21 minuti e 32 secondi.
LINEA 9020: fa terminare l'esecuzione del programma.

TIME$ (Variabile di sistema)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Ritorna l'ora mediante una stringa di 8 caratteri nella forma HH:MM:SS dove HH è l'ora (da 00 a 23), MM sono i minuti (da 00 a 59) e SS sono i secondi (da 00 a 59).

Per impostare inizialmente l'ora è necessario utilizzare una istruzione col seguente formalismo:

Esempio:

```
  10 REM * INIZIO PROGRAMMA *
  20 TIME$ = "00 : 00 : 00"
  30 :
8990
9000 REM * FINE PROGRAMMA *
9010 PRINT TIME$
9020 END
```

Risultato:

LINEA 10: serve per documentare il programma.
LINEA 20: azzera l'orologio del sistema.
LINEE 30÷8990: contengono il resto del programma.
LINEA 9000: serve per documentare il programma.
LINEA 9010: stampa il valore contenuto nella variabile di sistema TIME$. Se, ad esempio, venisse stampato il seguente valore:
00:21:32
ciò significa che il programma, in totale, è durato 21 minuti e 32 secondi.
LINEA 9020: fa terminare l'esecuzione del programma.

CAPITOLO 9

FUNZIONI DEFINITE DALL'UTENTE

GENERALITA'
DEF FN
FN

GENERALITA'

Se si deve utilizzare ripetutamente una equazione numerica o alfanumerica, conviene definirla sotto forma di funzione. Per far ciò occorre utilizzare l'istruzione DEF FN.

La funzione definita in tal modo dall'utente viene poi richiamata in modo analogo alle funzioni di sistema, utilizzando la funzione FN.

La definizione di tali funzioni è valida solo per quel determinato programma, e deve quindi essere ridefinita in tutti gli altri eventuali programmi concatenati (a meno che gli altri programmi non siano concatenati al primo con l'utilizzo del comando/istruzione MERGE).

L'istruzione DEF FN deve essere eseguita prima che la funzione che essa definisce possa essere richiamata.

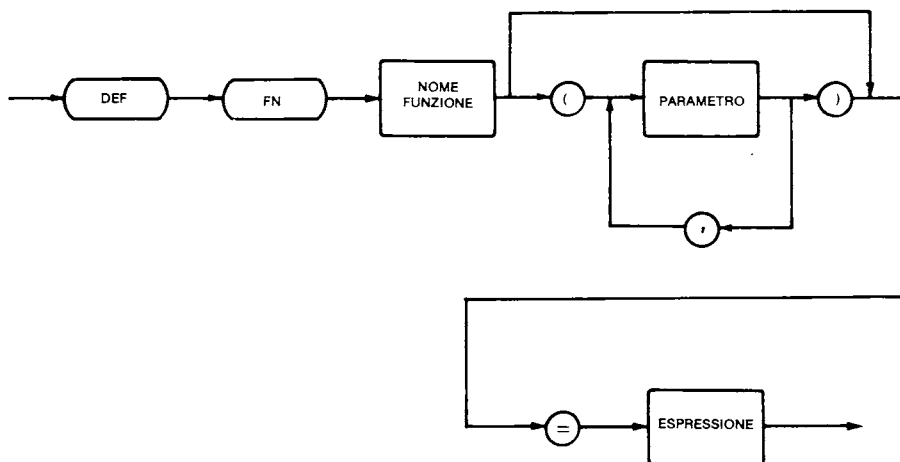
Una funzione può essere definita più di una volta. In tal caso verrà utilizzata la definizione più recente.

Si possono avere funzioni ricorrenti, che richiamano cioè se stesse. Tuttavia, se esse non forniscono una modalità per arrestare la ripetitività, si verificherà un errore per mancanza di memoria.

DEF FN (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Definisce una funzione di tipo numerico o stringa.

Formalismo sintattico:

Esempio:

```
10 PI = 3.141593
20 DEF FNA (R) = PI * R ^2
30 DEF FN V (R) = 4/3 * PI * R ^3
40 INPUT "DAMMI IL VALORE DEL RAGGIO"; R1
50 PRINT "L'AREA DEL CERCHIO È"; FNA (R1)
60 PRINT "IL VOLUME DELLA SFERA È"; FN V (R1)
```

Risultato:

LINEA 10: assegna il valore 3.141593 alla variabile numerica PI.

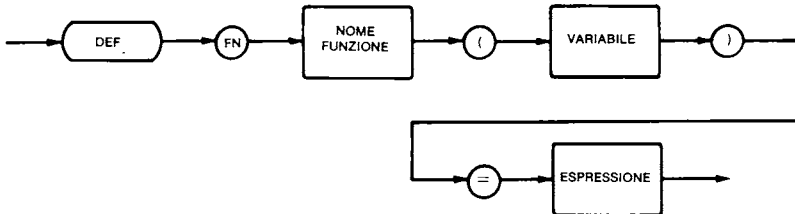
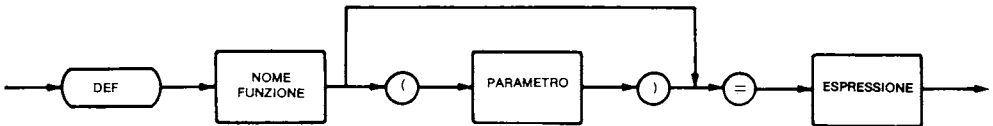
LINEA 20: definisce la funzione A che calcola l'area di un cerchio di raggio R. La variabile R è in questo caso il parametro della funzione.

LINEA 30: definisce la funzione V che calcola il volume della sfera di raggio R. Anche in questo caso la variabile R è il parametro della funzione.

LINEA 40: stampa il messaggio in oggetto e ferma l'esecuzione in attesa che venga introdotto un valore numerico da tastiera. Supponiamo di introdurre il valore 3. Esso verrà assegnato alla variabile R1.

LINEA 50: richiama la funzione A passando la variabile R1 come argomento. Il risultato sarà la seguente stampa:
L'AREA DEL CERCHIO È 28.274337

LINEA 60: richiama la funzione V passando il valore della variabile R1 come argomento. Il risultato sarà la stampa seguente:
IL VOLUME DELLA SFERA È 113.09734

Variante per: VIC 20 - CBM 64 - MZ 700 - APPLE

Variante per: TI 99/4A

Note:

— La variabile utilizzata come parametro (nel nostro caso la R) serve per definire la funzione. All'atto dell'utilizzo della funzione medesima, essa può essere sostituita da una costante numerica, da una variabile numerica (come la R1 dell'esempio) o anche da una espressione numerica.

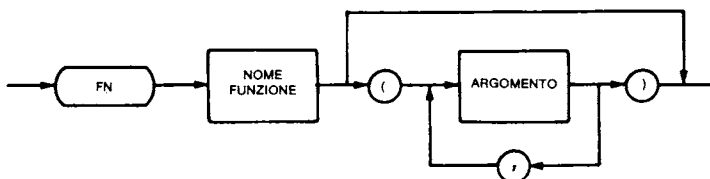
— Tutte le definizioni di funzione utente vengono annullate da una istruzione CLR o da un comando NEW.

— Non è possibile inserire una funzione nella definizione di una funzione utente.

FN (Istruzione)

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Serve per richiamare una funzione definita con un'istruzione DEF FN.

Formalismo sintattico:

Esempio:

```
10 DEF FN A (Y) = INT (Y * 100 + 0.5)/100
20 DEF FN B (Y) = INT (Y * 1000 + 0.5)/1000
30 INPUT "VALORE DA ARROTONDARE ="; L
40 PRINT "ARROTONDAMENTO AI DUE DECIMALI"; FN A (L)
50 PRINT "ARROTONDAMENTO AI TRE DECIMALI"; FN B (L)
```

Risultato:

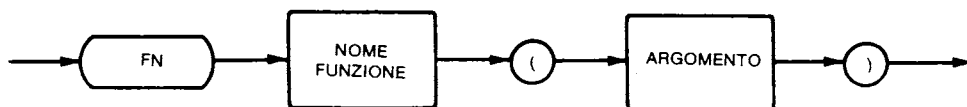
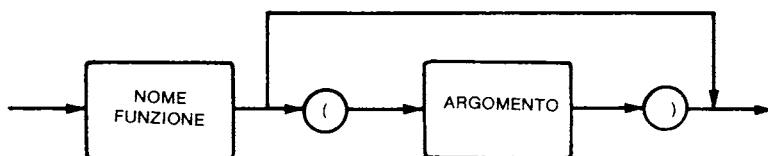
LINEA 10: Definisce la funzione A che arrotonda un numero ai primi due decimali. La variabile Y è in questo caso il parametro della funzione.

LINEA 20: Definisce la funzione B che arrotonda un numero alle prime tre cifre decimali. La variabile Y è in questo caso il parametro della funzione.

LINEA 30: Stampa il messaggio in oggetto e ferma l'esecuzione in attesa che venga introdotto un valore numerico da tastiera. Supponiamo di introdurre il vàlore 3.141593. Esso verrà assegnato alla variabile L.

LINEA 40: Richiama la funzione A passando la variabile L come argomento. Il risultato sarà la seguente stampa:
ARROTONDAMENTO AI DUE DECIMALI 3.14

LINEA 50: Richiama la funzione B passando la variabile L come argomento. Il risultato sarà la seguente stampa:
ARROTONDAMENTO AI TRE DECIMALI 3.142

Variante per: VIC 20 - CBM 64 - MZ 700 - APPLE

Variante per: TI 99/4A

Note:

— La variabile utilizzata come parametro nella definizione di una funzione (come la Y dell'esempio precedente) serve solo per definire la funzione stessa. All'atto dell'utilizzo essa può essere sostituita da una costante numerica, da una variabile numerica (come la L dell'esempio) o anche da una espressione numerica.

CAPITOLO 10

GRAFICA E COLORE

GENERALITA'

COMPORTAMENTO DEGLI ELABORATORI TRATTATI

TEXAS TI 99/4A
COMMODORE VIC 20
COMMODORE CBM 64
SINCLAIR ZX81
SINCLAIR ZX SPECTRUM
SHARP MZ-700
IBM P.C.
OLIVETTI M 20
APPLE II

ISTRUZIONI

CIRCLE
CALL HCHAR
CALL VCHAR
CALL GCHAR
CALL COLOR
CALL SCREEN
BORDER
COLOR
HCOLOR
PRINT []
PRINT [] USING
INK
PAPER
BRIGHT
FLASH
CONSOLE
DRAW
XDRAW
LINE
HLIN
VLIN
GET
PUT
GR
HGR
HGR2

SCREEN
INVERSE
NORMAL
OVER
PLOT
HPLOT
PAINT
PSET
PRESET
SET
UNPLOT
RESET
SCALE
TEXT
WINDOW (per generare una finestra)
WINDOW (per selezionare una finestra)
CLOSE WINDOW
WIDTH

FUNZIONI

ATTR
SCRN
SCREEN
SCREEN$
POINT
SCALEX
SCALEY
WINDOW

GENERAZIONE CARATTERI

GENERALITA'
TEXAS TI 99/4A
SINCLAIR ZX SPECTRUM
COMMODORE CBM 64
COMMODORE VIC 20
ISTRUZIONI
BIN
CALL CHAR

GENERAZIONE PROFILI

GENERALITA'
IBM P.C.
OLIVETTI M 20
APPLE II
ISTRUZIONI
ROT
SCALE
SHLOAD

GENERALITA'

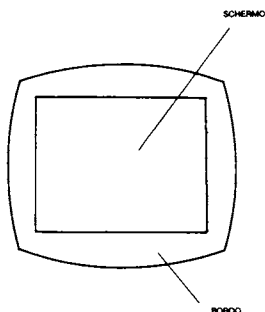
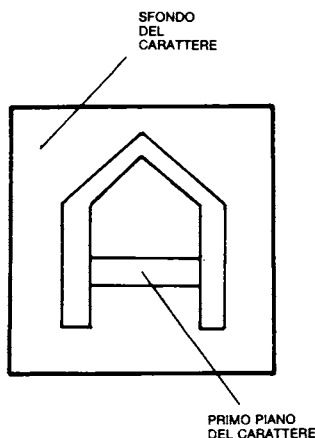
Tutti gli elaboratori funzionano in "modo testo", possono cioè visualizzare a video un certo numero di caratteri.

Alcuni di questi caratteri hanno caratteristiche grafiche, permettono cioè di costruire rettangoli, di visualizzare oggetti diversi da lettere, cifre e simboli speciali. In alcuni elaboratori è possibile per l'utente definire un certo numero di caratteri a suo piacimento (vedere "Generazione caratteri"). Questo modo di lavorare è detto "grafica a bassa risoluzione".

Alcuni elaboratori hanno la possibilità di lavorare anche in "grafica ad alta risoluzione". In tal modo lo schermo viene suddiviso in un numero (notevolmente più alto che nel "modo testo") di punti luminosi (detti "pixel", da "picture element"), ognuno dei quali può essere acceso, spento, colorato indipendentemente dagli altri. In tal modo è possibile visualizzare a video cerchi, linee rette o spezzate, funzioni di una variabile indipendente, ed altri oggetti che non sarebbe possibile visualizzare nel "modo testo".

In alcuni casi esistono due cursori indipendenti, uno per il "modo testo" e uno per il "modo grafico", in modo che sia possibile generare una combinazione di oggetti visualizzati nei due modi di funzionamento (per esempio, degli istogrammi o delle funzioni con anche la descrizione, in caratteri, del loro significato).

Per quanto riguarda il colore, è, in generale, possibile usufruire di 8 (o anche 16, in alcuni casi) colori diversi.

È anche normalmente possibile utilizzare colori diversi tra loro, e diversi da quelli previsti all'atto dell'accensione, per lo schermo video e per il bordo dello schermo stesso.

Per quanto riguarda i caratteri è anche possibile, in generale, colorare in modo diverso tra loro (e diverso anche da schermo e da bordo) lo sfondo di un carattere e il suo primo piano.

COMPORTAMENTO DEGLI ELABORATORI TRATTATI

TEXAS TI 99/4A

Esiste solo il "modo testo". Il video viene suddiviso in una matrice 25 x 32 (cioè 25 righe e 32 colonne), ed in tal modo è possibile visualizzare fino a 800 caratteri. Ogni elemento della matrice (cioè un carattere) è singolarmente indirizzabile, e può quindi venire, ad esempio, colorato in modo diverso dagli altri.

Si possono generare caratteri definiti dall'utente.

La gestione di tutte le possibilità grafiche del sistema è fattibile direttamente da BASIC, per mezzo di potenti sottoprogrammi richiamabili mediante CALL.

COMMODORE VIC 20

Esiste solo il "modo testo". Il video viene suddiviso in una matrice di 23 righe per 22 colonne, per cui è possibile visualizzare fino a 506 caratteri. Ogni carattere è singolarmente indirizzabile, e può quindi, ad esempio, venir colorato in modo diverso dagli altri.

È possibile, con un procedimento non banale, generare caratteri definiti dall'utente.

Esistono poche istruzioni BASIC orientate alla grafica: il tutto viene effettuato

con un utilizzo ossessivo dell'istruzione POKE e della funzione PEEK, il che rende i programmi grafici praticamente illeggibili e difficilmente convertibili su altri elaboratori, a meno che non si conosca esattamente la mappa di memoria del VIC 20.

COMMODORE CBM 64

Il "modo testo" è gestibile direttamente.

In tal caso il video viene diviso in una matrice di 25 righe e 40 colonne, per cui è possibile visualizzare fino a 1000 caratteri, indipendenti fra loro.

Mediante artifici, si può lavorare in grafica ad alta risoluzione (modo "BIT MAP"). In tal modo il video viene suddiviso in una matrice di 200 x 320 pixel, per cui è possibile visualizzare fino a 64.000 punti luminosi, indipendenti fra loro.

È anche possibile, con un procedimento non banale, generare caratteri definiti dall'utente e particolari profili (SPRITE).

Come per il VIC 20, esistono poche istruzioni BASIC orientate alla grafica: il tutto viene effettuato con un utilizzo ossessivo dell'istruzione POKE e della funzione PEEK, il che rende i programmi grafici praticamente illeggibili e difficilmente convertibili su altri elaboratori, a meno che non si conosca esattamente la mappa completa della memoria.

SINCLAIR ZX 81

Esiste sia il "modo testo" che il "modo grafico". Nel primo caso il video viene suddiviso in una matrice di 22 righe e 32 colonne, per cui è possibile visualizzare fino 704 caratteri, indipendenti fra loro. Nel secondo caso lo schermo viene suddiviso in una matrice 44 x 64 pixel, per cui è possibile visualizzare fino a 2.816 punti luminosi, indipendenti tra loro. Non è possibile generare caratteri definiti dall'utente. Non esistono istruzioni per la modifica del colore: lo ZX81 funziona solo in B/N.

SINCLAIR ZX SPECTRUM

Esistono sia il "modo testo" che il "modo grafico".

Nel primo caso il video viene suddiviso in una matrice di 22 righe per 32 colonne, per cui è possibile visualizzare fino a 704 caratteri, indipendenti fra loro. Nel "modo grafico" il video viene suddiviso in una matrice di 176 x 256 pixel, per cui si possono visualizzare fino a 44.800 punti luminosi, indipendenti fra loro.

È possibile generare caratteri definiti dall'utente.

Tutte le possibilità grafiche possono essere gestite direttamente dal BASIC, mediante opportune e potenti istruzioni.

Esiste sia il "modo testo" che il "modo grafico".

Nel primo caso il video viene suddiviso in una matrice di 25 righe e 40 colonne, per cui è possibile visualizzare fino a 1000 caratteri, indipendenti fra loro. Mediante l'istruzione CONSOLE si è in grado di definire una diversa area di scorrimento.

Nel "modo grafico" il video viene suddiviso in una matrice di 50 x 80 pixel, per cui è possibile visualizzare fino a 4.000 punti luminosi, indipendenti fra loro.

Non si possono generare caratteri definiti dall'utente.

In compenso l'MZ 700 ha un ulteriore set di 256 caratteri, tutti orientati alla grafica. La gestione della grafica è quasi completamente possibile attraverso potenti istruzioni BASIC. Solo in alcuni casi si ricorre all'uso dell'istruzione POKE e della funzione PEEK.

I.B.M. P.C.

Le possibilità di visualizzazione dipendono da quale adattatore per il video è stato installato.

Se è stato installato l'adattatore per l'unità video monocolore, si può lavorare solo in "modo testo" e in B/N. In tal modo il video viene suddiviso in 25 righe e 40 (oppure 80, se si è impostata un'ampiezza diversa con l'istruzione WIDTH) caratteri, per cui possono essere visualizzati fino a 1000 (2000) caratteri, indipendenti fra loro.

Se è stato installato l'adattatore di controllo del colore/grafici, è possibile lavorare in modo testo ma con 16 colori diversi (o anche in B/N). In più è possibile lavorare in "modo grafico". In tal caso si possono utilizzare due soluzioni:

- Grafica a media intensità: schermo suddiviso in 200 x 320 pixel, cioè fino a 64.000 punti luminosi, indipendenti fra loro. In tal modo si possono usare fino a 4 colori.
- Grafica ad alta intensità: schermo suddiviso in 200 x 640 pixel, cioè fino a 128.000 punti luminosi, indipendenti fra loro. In tal modo si possono usare fino a 2 colori.

Anche nei due modi grafici è possibile utilizzare i caratteri di testo. Le coordinate dei pixel possono essere date in forma assoluta, rispetto cioè a un punto fisso di riferimento, oppure in forma relativa, rispetto cioè alla posizione precedente del cursore grafico. Non è possibile generare caratteri definiti dall'utente.(*) Tutte le funzioni grafiche sono gestibili direttamente dal BASIC, mediante potenti istruzioni apposite.

(*) Vedere però il paragrafo "GENERAZIONE PROFILI".

OLIVETTI M 20

È possibile lavorare sia in "modo testo" che in "modo grafico". Nel primo caso il video viene suddiviso in 16 righe e 64 colonne, per un totale di 1024 caratteri, indipendenti fra loro. Via software è possibile portare tale suddivisione fino a 25 righe e 80 colonne, per un totale di 2000 caratteri, indipendenti fra loro.

In "modo grafico" si può suddividere il video in una matrice di 256 x 512 pixel (131.072 punti luminosi, corrispondente al testo 16 x 64) oppure in una matrice di 256 x 480 pixel (122.880 punti luminosi, corrispondente al testo 25 x 80). Il video può essere suddiviso dall'utente in aree rettangolari chiamate "finestre" ("windows").

Su ogni finestra l'utente può operare esattamente come sull'intero video: su esse può eseguire operazioni di grafica, di testo, oppure di entrambe. Le operazioni eseguite su finestre diverse sono completamente indipendenti. Se si lavora in modo grafico, l'utente può definire un proprio sistema di coordinate, valido solo per quella finestra.

Esistono due cursori: un cursore testo (la cui posizione è modificabile mediante l'istruzione CURSOR) e un cursore grafico (la cui posizione è modificabile mediante l'istruzione CURSOR POINT). È anche possibile, mediante comandi PCOS richiamabili da BASIC con le istruzioni CALL o EXEC, visualizzare stringhe di caratteri di dimensioni e orientamento variabili.

Non è possibile generare caratteri definiti dall'utente. (*)

Tutte le funzioni grafiche sono gestibili direttamente dal BASIC, mediante potenti istruzioni apposite.

(*) Vedere però il paragrafo "GENERAZIONE PROFILI"

APPLE II

È possibile lavorare sia in "modo testo" che in "modo grafico". Nel primo caso il video viene suddiviso in 24 righe e 40 colonne per un totale di 960 caratteri, indipendenti fra loro. Esistono poi due modi grafici. Il primo è detto "a bassa risoluzione" (definito dall'istruzione GR) in cui il video è suddiviso in 40 righe e 40 colonne, più 4 righe di testo alla base. È possibile, a schermo pieno, arrivare fino a 48 righe e 40 colonne, per un totale di 1920 grossi punti luminosi. Vi è poi il modo "ad alta risoluzione", in cui il video può essere suddiviso in una matrice di 280 x 160 pixel (44.800 pixel) utilizzando l'istruzione HGR, oppure in una matrice di 280 x 192 pixel (53.760 pixel) lavorando a schermo pieno, richiesto con l'istruzione HGR2. La gestione della grafica è quasi completamente possibile attraverso potenti istruzioni BASIC. Solo in alcuni casi si ricorre all'uso dell'istruzione POKE e della funzione PEEK.

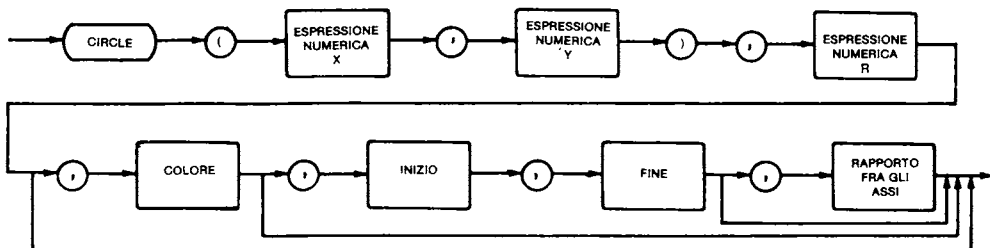
Non è possibile generare caratteri definiti dall'utente, ma si può invece, nel modo grafico, generare figure particolari (vedere paragrafo "GENERAZIONE PROFILI").

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

CIRCLE (Istruzione)

Traccia una circonferenza o un'ellisse. Le coordinate "X", "Y" individuano il centro, il parametro "R" individua il raggio della circonferenza oppure il semiasse orizzontale dell'ellissi.

Nello ZX Spectrum disegna una circonferenza di centro (X, Y) e di raggio R.

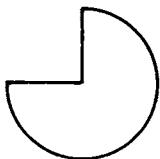
Formalismo sintattico:

Esempio:

```
10 PI = 3.141593
20 SCREEN 1 : CLS
30 CIRCLE (160, 100), 60,, – PI, – PI/2
40 FOR I = 1 TO 10000 : NEXT I : CLS
50 CIRCLE (20, 20), 19, 2
```

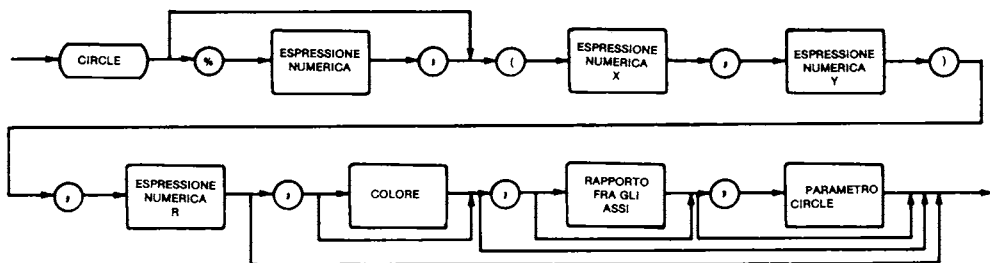
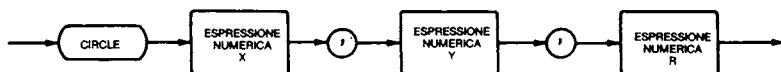
Risultato:

LINEA 10: assegna alla variabile PI il valore specificato (PI greco).

LINEA 20: imposta il modo video a media risoluzione e pulisce lo schermo.

LINEA 30: disegna un arco di cerchio da 180° a 90° in senso antiorario, di centro (160, 100) e raggio 60, con i vertici collegati al centro, e cioè:

LINEA 40: esegue un ciclo di ritardo per permettere la visualizzazione dell'oggetto stampato, e poi pulisce il video.

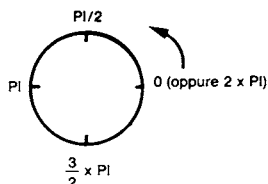
LINEA 50: disegna un cerchio completo con centro alle coordinate (20,20), di raggio 19 e di colore magenta.

Variante per: M 20

Variante per: ZX Spectrum

Note:

— ESPRESSIONE NUMERICA (in M20) indica la finestra sulla quale si vuol disegnare il cerchio, e può assumere valori compresi tra 1 e 16.

— ESPRESSIONE NUMERICA X, ESPRESSIONE NUMERICA Y, ESPRESSIONE NUMERICA R, identificano, rispettivamente, la coordinata X e la coordinata Y del raggio e la lunghezza del raggio stesso. I possibili valori assunti dipendono dal numero di pixel in cui è stato suddiviso il video (vedere il paragrafo "generalità" di questo stesso capitolo).

— COLORE è un'espressione numerica con valori compresi tra 0 e 3 per IBM P.C. e tra 0 e 7 per M 20. Vedere l'istruzione COLOR per maggiori dettagli.

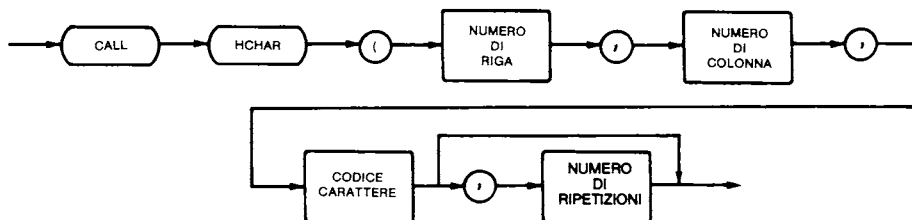
— INIZIO e FINE sono espressioni numeriche e possono assumere valori compresi tra −6.283186 e +6.283186. Individuano il punto di inizio e quello di fine del cerchio (o dell'ellisse).

— RAPPORTO FRA GLI ASSI è un'espressione numerica che determina l'aspetto dell'ellisse che verrà disegnata. Se il valore dell'espressione è minore di uno, allora il valore degli assi sarà uguale a R per l'asse delle X e a R*N per l'asse delle Y, dove N è il valore assunto da RAPP. FRA GLI ASSI. Se N è maggiore di uno, allora il valore degli assi sarà uguale a R per l'asse delle Y e a R*N per l'asse delle X.

— PARAMETRO CIRCLE individua una serie di possibilità opzionali. Vedere l'allegato A per l'elenco di tutte le opzioni.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CALL HCHAR (Istruzione)

Nel TEXAS TI 99/4A il sottoprogramma HCHAR stampa un carattere in qualsiasi posizione del video, ed eventualmente lo ripete un dato numero di volte sulla stessa linea.

Formalismo sintattico:

Esempio:

```
100 CALL HCHAR (15, 7, 65, 20)
110 FOR I = 1 TO 500
120 NEXT I
130 CALL CLEAR
140 FOR J = 10 TO 14
150 CALL HCHAR (J, 14, 90, 5)
160 NEXT J
170 END
```

Risultato:

LINEA 100: stampa il carattere A partendo dalla riga 15, colonna 7 e ripetendolo 20 volte in orizzontale.
LINEE 110÷120: ciclo di ritardo per permettere la visualizzazione.
LINEA 130: pulisce il video.
LINEA 140: inizio di un ciclo ripetuto 5 volte.
LINEA 150: stampa del carattere Z, partendo dalla riga 10, colonna 14, ripetuto 5 volte in orizzontale. Ad ogni ripetizione si scende di una riga. Il risultato a fine ciclo è la stampa di un quadrato pieno di Z nel mezzo del video, con lato 5 caratteri.
LINEA 160: fine del ciclo di stampa.
LINEA 170: fa terminare l'esecuzione del programma.

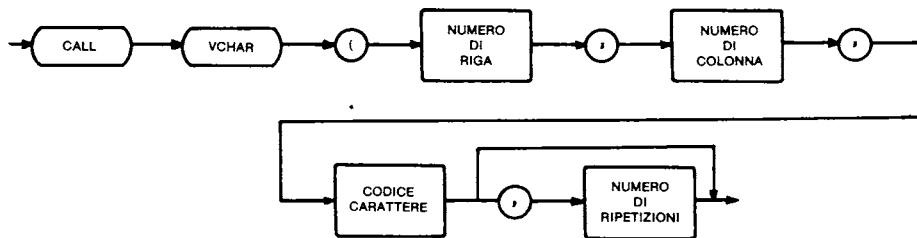
Note:

— NUMERO DI RIGA è un'espressione numerica che può assumere valori da 1 a 24.
— NUMERO DI COLONNA è un'espressione numerica che può assumere valori da 1 a 32 (però visualizza solo da 3 a 30).
— CODICE CARATTERE è un'espressione numerica che può assumere valori da 0 a 32767, anche se ha senso solo fino a 255. Valori maggiori di 255 verranno convertiti in valori compresi tra 0 e 255.
— NUMERO DI RIPETIZIONI è un'espressione numerica che può assumere valori da 0 a 32767.

CALL VCHAR (Istruzione)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Nel TI 99/4A il sottoprogramma VCHAR stampa un carattere in qualsiasi posizione del video, ed eventualmente lo ripete un dato numero di volte sulla stessa colonna.

Formalismo sintattico:

Esempio:

```
100 CALL VCHAR (15, 7, 65, 8)
110 FOR I = 1 TO 500
120 NEXT I
130 CALL CLEAR
140 FOR J = 14 TO 18
150 CALL VCHAR (10, J, 90, 5)
160 NEXT J
170 END
```

Risultato:

LINEA 100: stampa il carattere A partendo dalla riga 15, colonna 7 e lo ripete 8 volte in verticale.

LINEE 110÷120: ciclo di ritardo per permettere a visualizzazione.

LINEA 130: pulisce il video.

LINEA 140: inizio di un ciclo ripetuto 5 volte.

LINEA 150: stampa del carattere Z, partendo dalla riga 10 e colonna 14, ripetuto 5 volte in verticale. Ad ogni ripetizione del ciclo la stampa si sposta a destra di una colonna. Il risultato finale è la stampa di un quadrato pieno di Z nel centro del video, con lato 5 caratteri, identico a quello che viene stampato nell'esempio riportato sotto l'istruzione CALL HCHAR.

LINEA 160: fine del ciclo di stampa.

LINEA 170: fa terminare l'esecuzione del programma.

Note:

— NUMERO DI RIGA è un'espressione numerica che può assumere valori da 1 a 24.

— NUMERO DI COLONNA è un'espressione numerica che può assumere valori da 1 a 32 (però visualizza solo da 3 a 30).

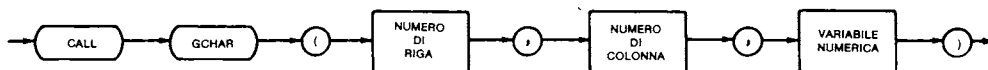
— CODICE CARATTERE è un'espressione numerica che può assumere valori da 0 a 32767, anche se ha senso solo fino a 255. Valori maggiori di 255 verranno convertiti in valori compresi tra 0 e 255.

— NUMERO DI RIPETIZIONI è un'espressione numerica che può assumere valori tra 0 e 32767.

CALL GCHAR (Istruzione)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Nel TI 99/4A il sottoprogramma GCHAR serve per leggere un carattere da qualsiasi posizione del video, individuata dal <numero di riga> e dal <numero di colonna>.

Formalismo sintattico:

Esempio:

```
100 CALL CLEAR
110 FOR I = 1 TO 24
120 CALL HCHAR (I, 1, 42, 32)
130 NEXT I
140 CALL GCHAR (10, 15, A)
150 CALL CLEAR
160 PRINT "INTERCETTATO IL CARATTERE"; CHR$ (A)
170 END
```

Risultato:

LINEA 100: pulisce il video.

LINEE 110÷130: con un ciclo da 1 a 24 (per le righe) e con ripetizione dalla colonna 1 alla 32, riempie il video di asterischi (il cui cosice ASCII è 42).

LINEA 140: legge il carattere presente sul video alla riga 10 e colonna 15; il codice ASCII di tale carattere viene memorizzato nella variabile A.

LINEA 150: pulisce il video.

LINEA 160: stampa, di fianco al messaggio specificato, il carattere intercettato dalla CALL GCHAR, e quindi:
INTERCETTATO IL CARATTERE *

Note:

— NUMERO DI RIGA è una espressione numerica che può assumere valori compresi fra 1 e 24.

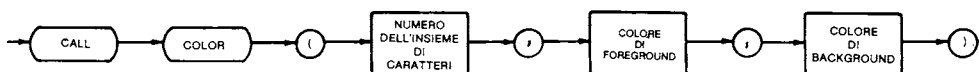
— NUMERO DI COLONNA è una espressione numerica che può assumere valori compresi fra 1 e 32 (però viene visualizzato solo il campo da 3 a 30).

— Nella variabile numerica viene memorizzato il codice ASCII del carattere letto dal video.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CALL COLOR (Istruzione)

Nel TI 99/4A il sottoprogramma COLOR permette di specificare il colore di primo piano e il colore di sfondo dei caratteri (a gruppi di 8).

Formalismo sintattico:

Esempio:

```
100 CALL CLEAR
110 PRINT "COLORE DEI CARATTERI:"
120 PRINT
130 INPUT "PRIMO PIANO:" : P
140 INPUT "SFONDO:" : S
150 CALL COLOR (5, P, S)
160 CALL CLEAR
170 CALL HCHAR (10, 15, 65, 4)
180 FOR I = 1 TO 500
190 NEXT I
200 END
```

Risultato:

LINEA 100: pulisce il video.

LINEE 110÷120: stampano il messaggio specificato ed una riga a spazi.

LINEE 130÷140: stampano il messaggio specificato e richiedono all'utente dei valori numerici da tastiera; i numeri digitati vengono poi memorizzati, rispettivamente, nelle variabili numeriche P e S. Si supponga che i valori digitati siano 2 e 14.

LINEA 150: specifica il colore di primo piano uguale al contenuto di P, e cioè nero, e il colore di sfondo uguale al contenuto di S, e cioè magenta, per il quinto insieme di caratteri, cioè per tutti i caratteri che hanno codice ASCII compreso tra 64 e 71.

LINEA 160: pulisce il video.

LINEA 170: stampa il carattere A alla riga 10, colonna 15, e lo ripete quattro volte in orizzontale. I 4 caratteri verranno stampati in nero su sfondo magenta.

LINEE 180÷190: ciclo di ritardo per permettere la visualizzazione dei caratteri stampati.

LINEA 200: fa terminare l'esecuzione del programma. Dopo l'esecuzione di questa istruzione, i colori torneranno ai valori standard, e cioè nero per il primo piano (foreground) e trasparente per lo sfondo (background).

Note:

— NUMERO DELL'INSIEME DI CARATTERI è una espressione numerica che può assumere i seguenti valori:

1 per codici ASCII da 32 a 39
2 per codici ASCII da 40 a 47
3 per codici ASCII da 48 a 55
4 per codici ASCII da 56 a 63
5 per codici ASCII da 64 a 71
6 per codici ASCII da 72 a 79
7 per codici ASCII da 80 a 87
8 per codici ASCII da 88 a 95
9 per codici ASCII da 96 a 103
10 per codici ASCII da 104 a 111
11 per codici ASCII da 112 a 119
12 per codici ASCII da 120 a 127
13 per codici ASCII da 128 a 135
14 per codici ASCII da 136 a 143
15 per codici ASCII da 144 a 151
16 per codici ASCII da 152 a 159

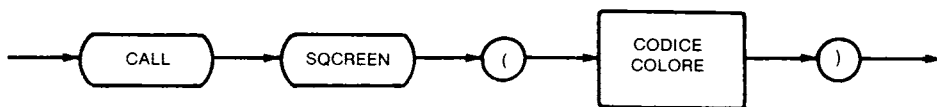
— COLORE DI FOREGROUND e COLORE DI BACKGROUND sono espressioni numeriche che possono assumere i seguenti valori:

1 Trasparente
2 Nero
3 Verde
4 Verde chiaro
5 Blu scuro
6 Blu chiaro
7 Rosso scuro
8 Viola
9 Rosso
10 Rosso chiaro
11 Giallo scuro
12 Giallo chiaro
13 Verde scuro
14 magenta
15 grigio
16 Bianco

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CALL SCREEN (Istruzione)

Nel TI 99/4A il sottoprogramma SCREEN permette di cambiare il colore dello schermo (il colore standard è verde chiaro).

Formalismo sintattico:

Esempio:

```
100 CALL CLEAR
110 PRINT "SCELTA DEI COLORI"
120 PRINT
130 INPUT "SCHERMO VIDEO" : V
140 INPUT "PRIMO PIANO" : P
150 INPUT "SFONDO" : S
160 CALL SCREEN (V)
170 CALL COLOR (5, P, S)
180 CALL CLEAR
190 CALL HCHAR (10, 15, 65, 4)
200 FOR I = 1 TO 500
210 NEXT I
220 END
```

Risultato:

LINEA 100: pulisce il video.

LINEE 110÷120: stampano il messaggio specificato ed una riga a spazi.

LINEE 130÷150: stampano il messaggio specificato e richiedono all'utente dei valori numerici da tastiera; i numeri digitati vengono poi memorizzati, rispettivamente, nelle variabili numeriche V, P e S. Si supponga che i valori digitati siano 7, 13 e 16.

LINEA 160: cambia il colore dello schermo in rosso scuro.

LINEA 170: specifica che il primo piano dei caratteri sarà verde scuro e lo sfondo dei caratteri sarà bianco.

LINEA 180: pulisce il video.

LINEA 190: stampa il carattere A alla riga 10 e alla colonna 15 e lo ripete quattro volte in orizzontale. I quattro caratteri verranno stampati in verde scuro su sfondo bianco su schermo rosso scuro.

LINEE 200÷210: ciclo di ritardo per permettere la visualizzazione dei caratteri stampati.

LINEA 220: fa terminare l'esecuzione del programma. Dopo l'esecuzione di questa istruzione, i colori torneranno ai valori standard, e cioè verde chiaro per lo schermo, nero per il primo piano dei caratteri e trasparente per lo sfondo dei caratteri.

Note:

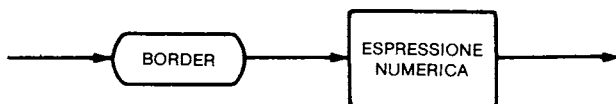
— CODICE COLORE è una espressione numerica che può assumere i seguenti valori:

1 Trasparente
2 Nero
3 Verde
4 Verde chiaro
5 Blu scuro
6 Blu chiaro
7 Rosso scuro
8 Viola
9 Rosso
10 Rosso chiaro
11 Giallo scuro
12 Giallo chiaro
13 Verde scuro
14 magenta
15 grigio
16 Bianco

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

BORDER (Istruzione)

Nello ZX Spectrum determina il colore per il bordo dello schermo, che è anche il colore dello sfondo per la parte bassa dello schermo.

Formalismo sintattico:

Esempio:

```
10 BORDER 2
20 PAPER 6
30 INK 1
40 PRINT "CARATTERI BLU SU SFONDO GIALLO E
   BORDO ROSSO"
```

Risultato:

LINEA 10: assegna il colore rosso al bordo dello schermo.
LINEA 20: assegna il colore giallo allo sfondo del carattere.
LINEA 30: assegna il colore blu al primo piano del carattere.
LINEA 40: stampa il messaggio specificato nei colori assegnati.

Note:

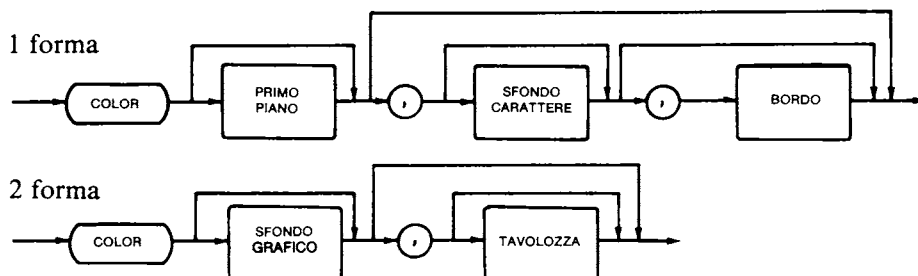
— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 7 che assumono il seguente significato:

0 Nero
1 Blu
2 Rosso
3 Porpora (o magenta)
4 Verde
5 Ciano (blu pallido)
6 giallo
7 Bianco

COLOR (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Per I.B.M. P.C. imposta i colori per il primo piano, lo sfondo e il bordo dello schermo. La forma 1 è valida quando si è in modo testo, la forma 2 quando si è in modo grafico.

Formalismo sintattico per: I.B.M. P.C.

Esempio:

In modo testo:

```
10 COLOR 1, 14, 4
20 CLS
30 PRINT "CARATTERI BLU SU SFONDO GIALLO CON
   BORDO SCHERMO ROSSO"
```

In modo grafico:

```
10 SCREEN 1 : CLS
20 COLOR 14, 0
30 CIRCLE (100, 100), 40, 2
```

Risultato: in modo testo:

LINEA 10: imposta il blu per il primo piano dei caratteri, il giallo per lo sfondo dei caratteri e il rosso per il bordo dello schermo.

LINEA 20: pulisce il video..

LINEA 30: stampa il messaggio specificato con i colori impostati.

In modo grafico:

LINEA 10: imposta il modo grafico a media risoluzione e pulisce lo schermo video.

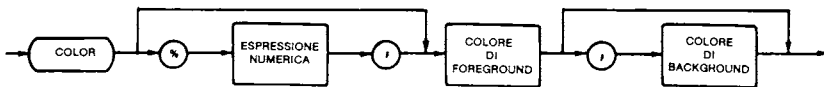
LINEA 20: imposta il giallo per lo sfondo e sceglie la tavolozza 0.

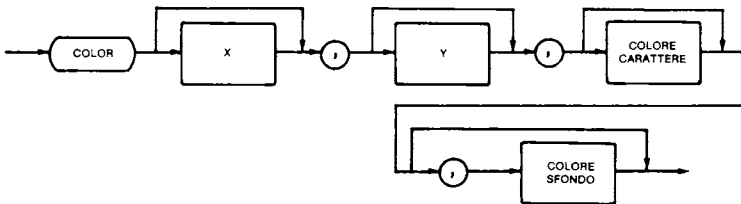
LINEA 30: disegna un cerchio alle coordinate (100,100), di raggio 40 e di colore rosso.

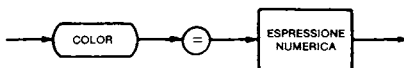
Per M 20 seleziona i quattro colori contemporanei tra gli otto possibili. L'espressione numerica dovrà avere valori compresi tra 0 e 7. Nella seconda forma seleziona i colori di background e foreground per una finestra specificata.

Formalismo sintattico per: M 20

1ª forma

2ª forma

Per MZ 700 viene usata per determinare i colori di primo piano (foreground) e di sfondo (background) per i caratteri, in una specifica posizione dello schermo.

Formalismo sintattico per: MZ 700:

Per Apple definisce il colore per il tracciamento nel modo per grafici a bassa risoluzione.

Formalismo sintattico per: Apple

Note:
Data la notevole differenza tra i diversi sistemi è opportuno procedere ad una distinzione.

— PRIMO PIANO è una espressione numerica con valori compresi tra 0 e 31, che imposta il colore di primo piano dei caratteri secondo la seguente tabella:

0 Nero
1 Blu
2 Verde
3 Ciano
4 Rosso
5 Magenta
6 Marrone
7 Bianco
8 Grigio
9 Blu chiaro
10 Verde chiaro
11 Ciano chiaro
12 Rosso chiaro
13 Magenta chiaro
14 Giallo
15 Bianco ad alta intensità

Aggiungendo il valore 16 a quelli visti è possibile far lampeggiare i caratteri
— SFONDO CARATTERE è un'espressione numerica con valori compresi tra 0 e 7, che imposta il colore di sfondo dei caratteri (primi otto elementi della tabella precedente).
— BORDO è un'espressione numerica con valori compresi tra 0 e 15 (vedere tabella precedente) che imposta il colore del bordo dello schermo.
— SFONDO GRAFICO è una espressione numerica con valori compresi tra 0 e 15 (vedi tabella precedente), che imposta il colore di sfondo dello schermo quando si è in modo grafico.
— TAVOLOZZA è una espressione numerica con valori compresi tra 0 e 1. Quando, in una successiva istruzione grafica (CIRCLE, DRAW, ecc.) si sceglierà un colore (tra 1 e 3), esso dipenderà dal valore di TAVOLOZZA impostato nella precedente COLOR; più precisamente si avrà:

Colore	Tavolozza 0	Tavolozza 1
1	Verde	Ciano
2	Rosso	Magenta
3	Marrone	Bianco

OLIVETTI M20

— Le quattro espressioni numeriche della 1° Forma (ESPRESS. NUMERICA 0, 1, 2 e 3) possono assumere valori compresi tra 0 e 7, col seguente significato:

0 Nero
1 Verde
2 Rosso
3 Giallo
4 Blu
5 Ciano
6 Magenta
7 Bianco

Nelle successive istruzioni grafiche si farà riferimento ai quattro colori prescelti in base alla loro posizione (0, 1, 2 o 3).
— ESPRESSIONE NUMERICA (2° Forma) individua la finestra su cui operare, e può assumere valori compresi fra 1 e 16.
— COLORE DI FOREGROUND è una espressione numerica che può assumere valori compresi tra 0 e 3 e fa riferimento ai colori scelti nella istruzione COLOR 1° Forma. Imposta il colore di primo piano dei caratteri o dei grafici per quella finestra.
— COLORE DI BACKGROUND è una espressione numerica che può assumere valori compresi tra 0 e 3 e fa riferimento ai colori scelti nella istruzione COLOR 1° Forma. Imposta il colore di sfondo dei caratteri. Se omesso, viene assunto il valore impostato nella precedente istruzione COLOR (2° Forma).

APPLE II
— ESPRESSIONE NUMERICA può assumere valori compresi tra 0 e 255, che vengono trattati a gruppi di 16 (modulo 16) col seguente significato:

0 Nero
1 Magenta
2 Blu scuro
3 Porpora
4 Verde scuro
5 Grigio
6 Blu medio
7 Azzurro
8 Marrone
9 Arancio
10 Grigio
11 Rosa

12 Verde
13 Giallo
14 Grigio-azzurro
15 Bianco

SHARP MZ-700

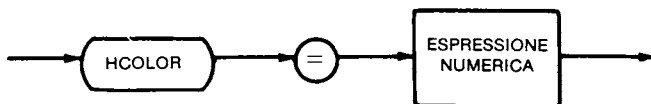
— X è una espressione numerica che può assumere valori compresi fra 0 e 39. Rappresenta la coordinata X del carattere interessato.
— Y è un'espressione numerica che può assumere valori compresi fra 0 e 24. Rappresenta la coordinata Y del carattere interessato.
— COLORE CARATTERE è un'espressione numerica che può assumere valori compresi fra 0 e 7, col seguente significato:

0 Nero
1 Blu
2 Rosso
3 Porpora
4 Verde
5 Azzurro
6 Giallo
7 Bianco

Identifica il colore di primo piano del carattere specificato.
— COLORE SFONDO è un'espressione numerica che può assumere valori compresi fra 0 e 7 (vedere tabella precedente). Identifica il colore per lo sfondo del carattere specificato.
— Se X e Y vengono omesse, i colori specificati valgono per l'intero schermo.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

HCOLOR (Istruzione)

Determina il colore con cui verranno tracciati i grafici in alta risoluzione.

Formalismo sintattico:

Esempio:

```
10 HGR
20 HCOLOR = 5
30 DRAW 1 AT 100, 70
40 END
```

Risultato:

LA LINEA 10 cancella il video e lo predispone in alta risoluzione, modo grafico.
LA LINEA 20 definisce il colore usato.
LA LINEA 30 definisce l'inizio del profilo 1 all'intersezione delle coordinate 100 e 70, e lo disegna.

Note:

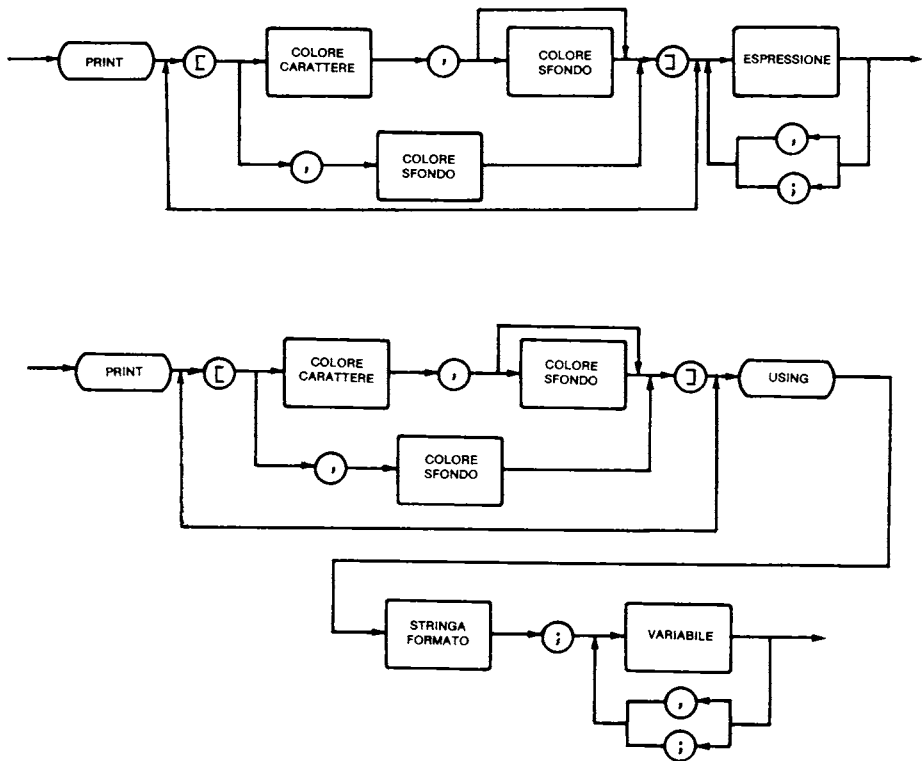
— L'espressione numerica deve assumere valori compresi fra 0 e 7, col seguente significato:

0 Nero 1
1 Verde (*)
2 Blu (*)
3 Bianco 1
4 Nero 2
5 (*)
6 (*)
7 Bianco 2

(*) Dipendono dal televisore utilizzato.

PRINT [ ]
PRINT [ ] USING (Istruzioni)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

In MZ-700 è possibile aggiungere delle specifiche di colore alle istruzioni PRINT e PRINT USING.
Esse possono quindi diventare le seguenti:

Esempio:

```
10 A$ = "TOPO" : B$ = "LINO" : C = 50
20 PRINT [ 6, 5 ] A$; B$
30 PRINT [ , 4 ] "HA";
40 PRINT [ , 5 ] USING "# # # #"; C,
50 PRINT [ , 4 ] "ANNI"
60 END
```

Risultato:

LINEA 10: assegna i valori specificati alle variabili A$, B$, C.

LINEA 20: stampa il contenuto delle due variabili stringa A$ e B$ ravvicinato, e cioè:
TOPOLINO
in giallo su sfondo azzurro.

LINEE 30÷50: stampano la seguente frase sul video:
HA 50 ANNI
in cui le parole "HA" e "ANNI" vengono stampate in giallo su sfondo verde, mentre le lettere 50 vengono stampate in giallo su sfondo azzurro.

LINEA 60: fa terminare l'esecuzione del programma. Le frasi stampate a video mantengono il loro colore. Le frasi successive verranno stampate con gli ultimi colori impostati, e cioè in giallo su sfondo verde.

Note:

— COLORE CARATTERE e COLORE SFONDO sono espressioni numeriche che possono assumere i seguenti valori:

0 Nero
1 Blu
2 Rosso
3 Porpora
4 Verde
5 Azzurro
6 Giallo
7 Bianco

— I caratteri di formattazione utilizzabili all'interno della stringa formato sono parecchi e diversificati nei vari dialetti. Rimandiamo alla istruzione PRINT USING e all'appendice A (alla voce STRINGA FORMATO) per la loro spiegazione.

INK (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Nello ZX Spectrum determina il colore di primo piano di tutti i caratteri stampati dopo la sua esecuzione (N = 0÷7 per un colore, N = 8 per la trasparenza, N = 9 per il contrasto).

Formalismo sintattico:

Esempio:

```
10 BORDER 2
20 PAPER 6
30 INK 1
40 PRINT "CARATTERI BLU SU SFONDO GIALLO E
   BORDO SCHERMO ROSSO"
```

Risultato:

LINEA 10: assegna il colore rosso al bordo dello schermo.
LINEA 20: assegna il colore giallo allo sfondo del carattere.
LINEA 30: assegna il colore blu al primo piano del carattere.
LINEA 40: stampa il messaggio specificato nei colori assegnati.

Note:

— ESPRESSIONE NUMERICA N può assumere valori compresi tra 0 e 9, col seguente significato:

0 Nero
1 Blu
2 Rosso
3 Porpora (o magenta)
4 Verde
5 Ciano (blu pallido)
6 Giallo
7 Bianco
8 Trasparenza
9 Contrasto

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

PAPER (Istruzione)

Nello ZX Spectrum determina il colore di sfondo di tutti i caratteri stampati dopo la sua esecuzione (N = 0 ÷ 7 per un colore, N = 8 per la trasparenza, N = 9 per il contrasto).

Formalismo sintattico:

Esempio:

```
10 BORDER 2
20 PAPER 6
30 INK 1
40 PRINT "CARATTERI BLU SU SFONDO GIALLO E
   BORDO SCHERMO ROSSO"
```

Risultato:
LINEA 10: assegna il colore rosso al bordo dello schermo.
LINEA 20: assegna il colore giallo allo sfondo del carattere.
LINEA 30: assegna il colore blu al primo piano del carattere.
LINEA 40: stampa il messaggio specificato nei colori assegnati.

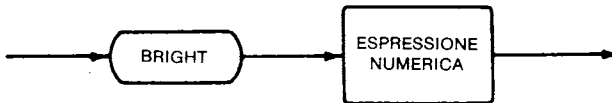
Note:
— ESPRESSIONE NUMERICA N può assumere valori compresi fra 0 e 9, col seguente significato:

0 Nero
1 Blu
2 Rosso
3 Porpora (o magenta)
4 Verde
5 Ciano (blu pallido)
6 Giallo
7 Bianco
8 Trasparenza
9 Contrasto

BRIGHT (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Nello ZX Spectrum determina la luminosità dei caratteri stampati dal momento della sua esecuzione in poi.

Formalismo sintattico:

Esempio:

```
10 FOR L = 0 TO 1
20 BRIGHT L
30 PAPER 2 : PRINT "PIPPO"
40 FOR N = 1 TO 1000 : NEXT N
50 NEXT L
```

Risultato:

LINEA 10: innesca un ciclo di due ripetizioni.
LINEA 20: determina una luminosità dipendente dalla ripetizione del ciclo: dapprima luminosità normale, la seconda volta extraluminosità.
LINEA 30: imposta il rosso come primo piano dei caratteri e stampa la parola specificata.
LINEA 40: ciclo di ritardo per permettere la visualizzazione.
LINEA 50: termine del ciclo iniziato alla linea 10.

Note:

— ESPRESSIONE NUMERICA può assumere i seguenti valori:
0 = luminosità normale
1 = extraluminosità
8 = trasparenza

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

FLASH (Istruzione)

Nello ZX Spectrum definisce se i caratteri stampati successivamente alla sua esecuzione devono essere lampeggianti o stabili.

Formalismo sintattico per: ZX Spectrum

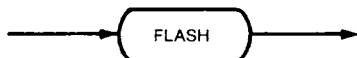
Esempio:

```
10 FOR L = 0 TO 1
20 FLASH L
30 PAPER 2 : PRINT "PIPPO"
40 FOR N = 1 TO 1000 : NEXT N
50 NEXT L
```

Risultato:

LINEA 10: innesca un ciclo di due ripetizioni.
LINEA 20: determina un diverso lampeggiamento ad ogni ripetizione: dapprima caratteri stabili, la seconda volta caratteri lampeggianti.
LINEA 30: imposta il rosso come primo piano dei caratteri e stampa la parola specificata.
LINEA 40: ciclo di ritardo per permettere la visualizzazione.
LINEA 50: termine del ciclo iniziato alla linea 10.

In Apple definisce il modo video lampeggiante, cosicchè l'output del computer viene alternativamente presentato sullo schermo in bianco su fondo nero e quindi presentato in nero su fondo bianco.

Formalismo sintattico per: Apple II

Note:

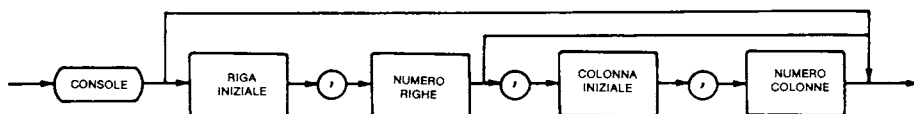
— ESPRESSIONE NUMERICA può assumere i seguenti valori:
0 = caratteri stabili

1 = caratteri lampeggianti
8 = nessun cambiamento.

CONSOLE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Questa istruzione specifica l'ampiezza dell'area di scorrimento, cioè dell'area che viene annullata da PRINT "C".

Formalismo sintattico:

Esempio:
1) CONSOLE 0, 25, 0, 40
2) CONSOLE 6, 12
3) CONSOLE 0, 25, 10, 25
4) CONSOLE 0, 7, 0, 7
5) CONSOLE

Risultati:
1) Specifica l'intero schermo come area di scorrimento.
2) Specifica l'area compresa fra la riga 6 e la riga 17 come area di scorrimento.
3) Specifica l'area compresa tra le colonne 10 e 34 come area di scorrimento.
4) Specifica il quadrato 7 x 7 nell'angolo superiore sinistro come area di scorrimento.
5) Non specifica l'area di scorrimento, per cui è possibile far scorrere lo schermo verso l'alto o verso il basso.

Note:
— RIGA INIZIALE è un'espressione numerica che può assumere valori da 0 a 24.
— NUMERO RIGHE è un'espressione numerica che può assumere valori da 1 a 25.
— COLONNA INIZIALE è un'espressione numerica che può assumere valori da 0 a 39.
— NUMERO COLONNE è un'espressione numerica che può assumere valori da 1 a 40.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

DRAW (Istruzione)

Traccia linee di un colore specificato (per M 20 all'interno della finestra specificata). È valido solo in modo grafico.

Per ZX Spectrum disegna una linea spostandosi orizzontalmente di X, verticalmente di Y, relativamente alla corrente posizione di PLOT e ruotando intorno ad un angolo Z. Se Z = 0 disegna una linea retta.
Per APPLE disegna (nel modo grafico ad alta risoluzione) un profilo (definito dall'espressione numerica 1) partendo dall'ultimo punto tracciato dal comando HPLOT (o DRAW o XDRAW) più recente oppure, se presente l'opzione AT, partendo dal punto con coordinata X, Y.

Formalismo sintattico:

Esempio:

```
10 SCREEN 1 : CLS
20 COLOR 0, 0
30 DRAW "C1BM12, 26L25E5BG5F5"
40 DIM DARK% (35)
50 GET (0, 20) – (18, 30), DARK%
```

Risultato:

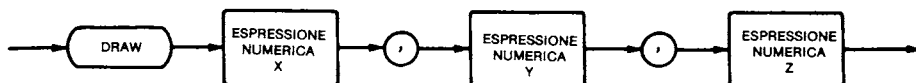
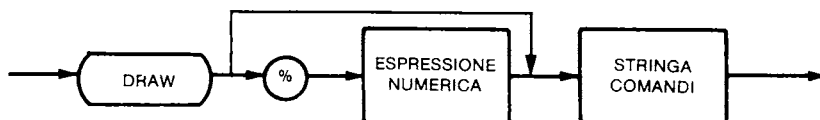
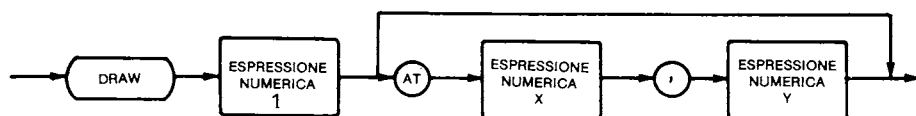
LINEA 10: imposta il modo a media risoluzione e pulisce lo schermo video.

LINEA 20: imposta i colori: nero per lo sfondo e tavolozza 0.

LINEA 30: disegna una freccia nel modo seguente: imposta dapprima il colore 1 (che, nella tavolozza 0 corrisponde al verde), si sposta, senza scrivere, alle coordinate (12, 26), si sposta a sinistra, scrivendo, di 25 pixel, si sposta in diagonale verso l'alto e a destra di 5 pixel, scrivendo, si sposta, senza scrivere, in diagonale verso il basso e a sinistra di 5 pixel e infine, scrivendo, si sposta in diagonale verso il basso e a destra di 5 pixel. Questo origina la seguente figura:

LINEA 40: dimensiona la matrice DARK%

LINEA 50: salva, nella matrice DARK%, il profilo della freccia appena costruito.

Variante per: ZX Spectrum

Variante per: M20

Variante per: APPLE

Note:

— STRINGA COMANDI (per IBM PC e Olivetti M20) identifica tutta una serie di comandi che costituiscono un vero e proprio linguaggio interno. Vedere l'appendice A per maggiori ragguagli (alla voce "STRINGA COMANDI").

— ESPRESSIONE NUMERICA (per M20) identifica la finestra interessata, e può assumere valori compresi fra 1 e 16.

— ESPRESSIONE NUMERICA X (per ZX Spectrum) può assumere valori compresi fra 0 e 255. Individua la coordinata orizzontale del pixel di arrivo.

— ESPRESSIONE NUMERICA Y (per ZX Spectrum) può assumere valori compresi fra 0 e 175. Individua la coordinata verticale del pixel di arrivo.

— ESPRESSIONE NUMERICA Z (per ZX Spectrum) può assumere valori compresi fra $-2\ \pi$ e $+2\ \pi$. Corrisponde alla frazione di cerchio che si vuol far disegnare.

— ESPRESSIONE NUMERICA 1 (per APPLE) identifica il profilo che si vuole disegnare, e può assumere valori compresi fra 0 e 255. Deve corrispondere ad un profilo effettivamente generato e memorizzato con tale valore.

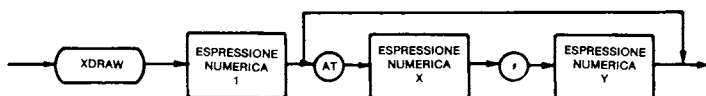
— ESPRESSIONE NUMERICA X (per APPLE) può assumere valori compresi fra 0 e 278.

— ESPRESSIONE NUMERICA Y (per APPLE) può assumere valori compresi fra 0 e 191.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

XDRAW (Istruzione)

In APPLE questa istruzione è identica a DRAW, tranne che il colore usato nel disegno del profilo è il complementare di quello già esistente in ciascun punto del profilo.

Formalismo sintattico:

Esempio:

```
10 HGR
20 ROT = 1 : SCALE = 2
30 XDRAW 1 AT 10, 100
```

Risultato:

LA LINEA 10 cancella il video e lo predispone in modo grafico in alta risoluzione.
LA LINEA 20 definisce l'angolo di rotazione e la scala.
LA LINEA 30 disegna il profilo 1 all'intersezione delle coordinate (10,100).

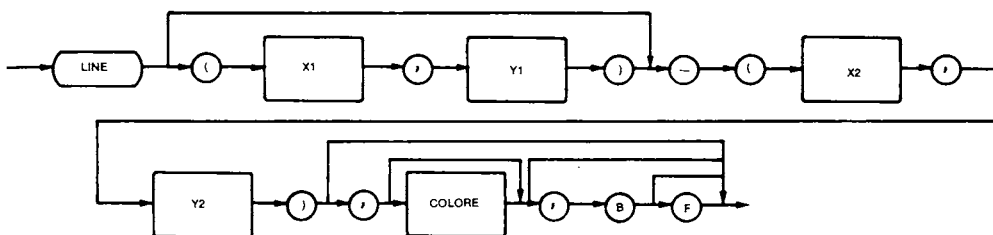
Note:

— ESPRESSIONE NUMERICA 1 compresa tra 0 e 255
— ESPRESSIONE NUMERICA X compresa tra 0 e 278
— ESPRESSIONE NUMERICA Y compresa tra 0 e 191
— Un utilizzo di XDRAW è, ad esempio, la cancellazione di un profilo disegnato in precedenza, nella identica posizione, con una istruzione DRAW.

LINE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Traccia una linea o un rettangolo in un determinato colore.

Formalismo sintattico per: I.B.M. P.C.

Esempio:

```
10 SCREEN 1 : CLS
20 COLOR 0, 0
30 LINE (10, 10) – (10, 100), 2
40 LINE (20, 10) – (50, 100), 2, B
50 LINE (70, 10) – (100, 100), 1, BF
60 END
```

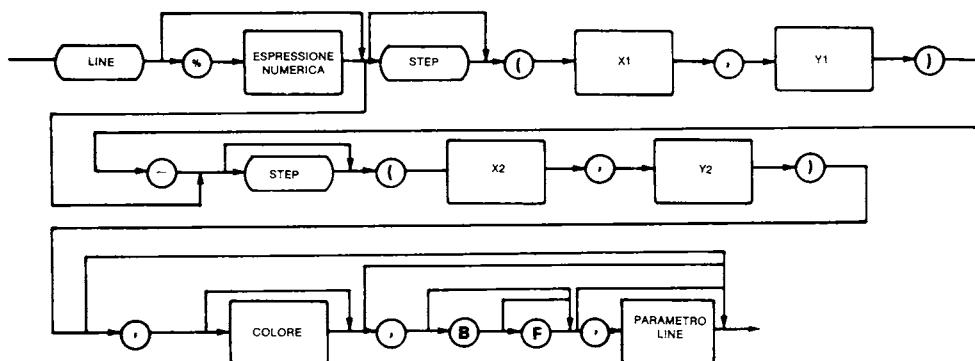
Risultato:

LINEA 10: imposta il modo grafico a media risoluzione e cancella lo schermo video.

LINEA 20: imposta il nero come colore di sfondo e sceglie la tavolozza 0.

LINEA 30: traccia una linea dal punto di coordinate (10,10) al punto di coordinate (10,100) di colore rosso.

LINEA 40: traccia un rettangolo che ha come vertici i punti di coordinate (20,10) (in alto a sinistra) e (50,100) (in basso a destra); le linee di cui è composto il rettangolo sono colorate in rosso.

LINEA 50: traccia un rettangolo che ha come vertici i punti di coordinate (70,10) (in alto a sinistra) e (100,100) in basso a destra); sia le linee di cui è composto il rettangolo che tutta l'area che esso delimita vengono colorate in verde.

Formalismo sintattico per: M 20

Note per I.B.M. P.C.:

— X1 e X2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 319.

— Y1 e Y2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 199.

— COLORE è un'espressione numerica che può assumere valori compresi fra 1 e 3.

Note per Olivetti M20:

— ESPRESSIONE NUMERICA può assumere valori compresi fra 1 e 16. Indica la finestra su cui si sta operando.

— X1 e X2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 511.

— Y1 e Y2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 255.

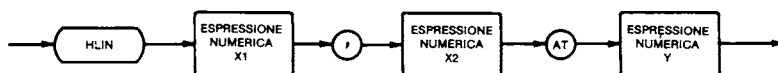
— COLORE è un'espressione numerica che può assumere valori compresi tra 0 e 3, e fa riferimento ai colori scelti nella specifica COLOR (1° Forma).

— PARAMETRO LINE individua una serie di possibilità opzionali. Vedere l'appendice A per l'elenco di tutte le opzioni.

HLIN (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Usata nel modo per grafici a bassa risoluzione, disegna una riga orizzontale da X1, Y a X2, Y. Il colore è determinato dall'istruzione COLOR eseguita più di recente.

Formalismo sintattico:

Esempio:

```
10 GR
20 COLOR = 5
30 HLIN 10, 20 AT 1
40 END
```

Risultato:

LA LINEA 10 cancella il video e lo predispone in modo grafico.
LA LINEA 20 definisce il colore (grigio).
LA LINEA 30 disegna una linea orizzontale dalla colonna 10 alla colonna 20 sulla riga 1.

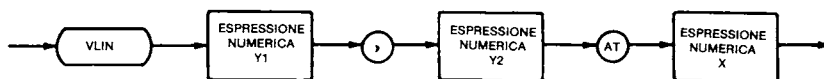
Note:

— X1 e X2 devono essere comprese tra 0 e 39, e Y tra 0 e 47.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

VLIN (Istruzione)

Usata nel modo per grafici a bassa risoluzione disegna una riga verticale daX, Y1 a X, Y2. Il colore è determinato dall'istruzione COLOR eseguita più di recente.

Formalismo sintattico:

Esempio:

```
10 GR
20 COLOR = 5
30 VLIN 0, 20 AT 10
40 END
```

Risultato:
LA LINEA 10 cancella il video.
LA LINEA 20 definisce il colore (grigio).
LA LINEA 30 disegna una linea dalla riga 0 alla riga 20 sulla colonna 10.

Note:
— X deve essere compreso tra 0 e 39, mentre Y1 e Y2 devono essere compresi tra 0 e 47.

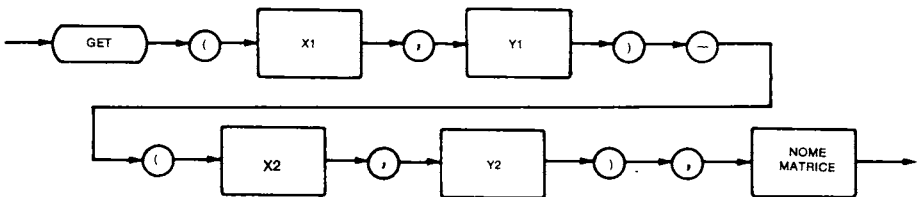
GET (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Nel M 20 memorizza l'intera finestra o una sua parte in una matrice intera unidimensionale. Per I.B.M. P.C. legge i punti da un'area dello schermo e li memorizza in una matrice.

Formalismo sintattico per I.B.M. P.C.:

Esempio:

```
10 SCREEN 1 : CLS : COLOR 0, 0
20 DRAW "C1BM12, 26L25E5BG5F5"
30 DIM DARK% (35)
40 GET (0, 20) – (18, 30), DARK%
50 CLS : I = 300
60 PUT (I, 100), DARK%, PSET
70 I = I – 1 : IF I < 0 THEN END
80 GOTO 60
```

Risultato:

LINEA 10: imposta il modo a media risoluzione, pulisce il video e imposta il colore.

LINEA 20: disegna il profilo di una freccia (punta a sinistra).

LINEE 30÷40: dimensionano la matrice DARK% e vi memorizzano il profilo della freccia appena generato.

LINEA 50: pulisce il videq e imposta la variabile numerica I.

LINEA 60: disegna il profilo della freccia alla coordinata verticale 100 e alla coordinata orizzontale variabile da 300 a 0, cancellando ogni volta l'immagine precedente. Ciò determina un movimento continuo della freccia.

LINEE 70÷80: diminuiscono di 1 il valore di I e ritornano ad eseguire la linea 60, a meno che la variabile I non sia diventata negativa, al che fanno terminare l'esecuzione del programma.

Formalismo sintattico per: M20

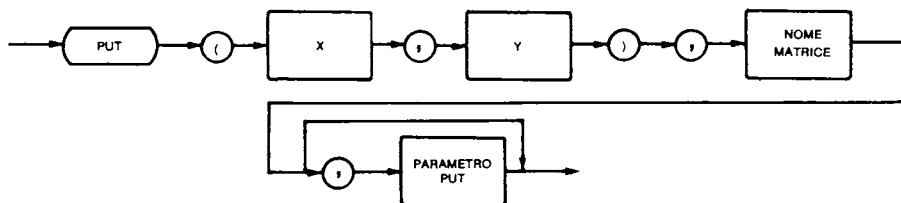
Note per I.B.M. P.C.:
- — X1 e X2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 319.
- — Y1 e Y2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 199.

Note per Olivetti M20:
- — ESPRESSIONE NUMERICA può assumere valori compresi tra 1 e 16, e individua la finestra su cui si sta lavorando.
- — X1 e X2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 511.
- — Y1 e Y2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 255.

PUT (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

In M20 visualizza un'immagine che era stata precedentemente memorizzata in una matrice intera unidimensionale (con l'istruzione GET). In I.B.M. P.C. scrive a video i dati contenuti in una matrice.

Formalismo sintattico per: I.B.M. P.C.

Esempio:

```
10 SCREEN 1 : CLS : COLOR 0, 0
20 DRAW "C1BM12, 26L25E5BG5F5"
30 DIM DARK% (35)
40 GET (0, 20) - (18, 30), DARK%
50 CLS : I = 300
60 PUT (I, 100), DARK%, PSET
70 I = I - 1 : IF I < 0 THEN END
80 GOTO 60
```

Risultato:

LINEA 10: imposta il modo a media risoluzione, pulisce il video e imposta il colore.

LINEA 20: disegna il profilo di una freccia con la punta rivolta a sinistra.

LINEE 30÷40: dimensionano la matrice DARK% e vi memorizzano il profilo della freccia appena generato.

LINEA 50: pulisce il video e imposta la variabile numerica I.

LINEA 60: disegna il profilo della freccia alla coordinata verticale 100 e alla coordinata orizzontale variabile da 300 a 0, cancellando ogni volta l'immagine precedente. Ciò determina un movimento continuo della freccia.

LINEE 70÷80: diminuiscono di 1 il valore di I e ritornano ad eseguire la linea 60, a meno che la variabile I non sia diventata negativa, al che fanno terminare l'esecuzione del programma.

Formalismo sintattico per M20

Note per I.B.M. P.C.:

— X è un'espressione numerica che può assumere valori compresi tra 0 e 319.
— Y è un'espressione numerica che può assumere valori compresi tra 0 e 199.
— PARAMETRO PUT individua una serie di possibilità opzionali. Vedere l'appendice A per l'elenco di tutte le opzioni.

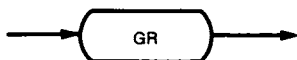
Note per Olivetti M 20:

— ESPRESSIONE NUMERICA può assumere valori compresi fra 1 e 16, e individua la finestra su cui si sta lavorando.
— X1 e X2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 511.
— Y1 e Y2 sono espressioni numeriche che possono assumere valori compresi fra 0 e 255.
— PARAMETRO PUT individua una serie di possibilità opzionali. Vedere l'ppendice A per l'elenco di tutte le opzioni.

GR (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Definisce sullo schermo il modo per grafici a bassa risoluzione (40 x 40 pixel), lasciando quattro righe per il testo alla base. Lo schermo viene oscurato e il cursore spostato alla finestra del testo.

Formalismo sintattico:

Esempio:

```
10 GR
20 COLOR = 5
30 HLIN 10, 20 AT 1
40 TEXT
50 END
```

Risultato:

LA LINEA 10 cancella il video e lo predispone in modo grafico.
LA LINEA 20 definisce il colore (grigio).
LA LINEA 30 disegna una linea orizzontale dalla colonna 10 alla colonna 20 sulla riga 1.
LA LINEA 40 ridefinisce il video nel modo testo a schermo intero.

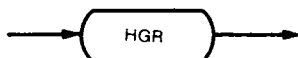
Note:

— Si può passare a grafici su schermo intero (40 x 80 pixel) con l'istruzione: POKE - 16302,0 oppure POKE 49234,0 dopo l'esecuzione dell'istruzione GR.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

HGR (Istruzione)

Definisce il modo per grafici ad alta risoluzione (280 x 160 pixel) per lo schermo, lasciando le ultime 4 righe per il testo.

Formalismo sintattico:

Esempio:

```
10 HGR
20 HCOLOR = 5
30 DRAW 1 AT 100, 70
40 END
```

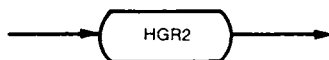
Risultato:

LA LINEA 10 cancella il video e lo predispone in alta risoluzione, modo grafico.
LA LINEA 20 definisce il colore usato.
30 definisce l'inizio del profilo 1 all'intersezione delle coordinate 100 e 70, e lo disegna.

HGR2 (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Definisce il modo per grafici ad alta risoluzione a schermo intero (280 x 192 pixel).

Formalismo sintattico:

Esempio:

```
10 HGR2
20 HCOLOR = 5
30 DRAW 1 AT 100, 70
40 END
```

Risultato:

LA LINEA 10 cancella il video e lo predispone in alta risoluzione, modo grafico, a schermo intero.

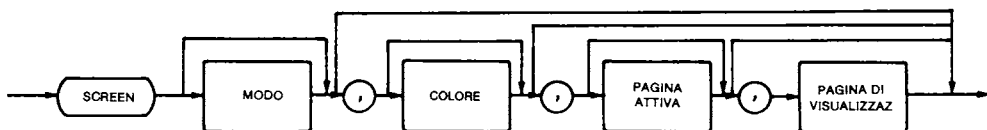
LA LINEA 20 definisce il colore usato.

LA LINEA 30 definisce l'inizio del profilo 1 all'intersezione delle coordinate 100 e 70, e lo disegna.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

SCREEN (Istruzione)

In I.B.M. P.C. definisce gli attributi dello schermo che devono essere utilizzati dalle istruzioni successive.

Formalismo sintattico:

Esempio:
1) SCREEN 0, 1, 0, 0
2) SCREEN 0, 1, 1, 2
3) SCREEN 1, 0
4) SCREEN 2,, 0, 0

Risultati:
Nell'esempio 1 viene scelto il modo testo, col colore e vengono impostate a zero le pagine attiva e di visualizzazione.
Nell'esempio 2 viene scelto il modo testo, col colore; la pagina attiva è impostata ad 1 e quella di visualizzazione a 2.
Nell'esempio 3 viene impostato il modo grafico a media risoluzione, abilitato al colore.
Nell'esempio 4 viene impostato il modo grafico ad alta risoluzione.

Note:
— MODO è un'espressione numerica che può assumere i seguenti valori:
 0 = modo testo
 1 = modo grafico a media risoluzione
 2 = modo grafico ad alta risoluzione
— COLORE è un'espressione numerica che può assumere i valori falso (0) e vero (diverso da zero): serve ad abilitare o disabilitare il colore.
— PAGINA ATTIVA e PAGINA DI VISUALIZZAZIONE sono espressioni numeriche che possono assumere valori da 0 a 7 (se il video è di 40 caratteri) o da 0 a 30 (se il video è di 80 caratteri). Sono valide solo in modo testo.

INVERSE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Nello ZX Spectrum controlla l'inversione dei caratteri stampati dopo la sua esecuzione.

In Apple definisce il modo video perchè l'output appaia in lettere nere su fondo bianco.

Formalismo sintattico per: ZX Spectrum

Esempio:

```
100 INVERSE 1
110 PRINT "PIPPO"
120 INVERSE 0
130 PRINT "PLUTO"
140 END
```

Risultato:

LA LINEA 100 predispone il sistema per la stampa in inverso (scambiando i colori di sfondo con quelli di primo piano, e viceversa).

LA LINEA 110 stampa: PIPPO (in modo inverso).

LA LINEA 120 ripristina l'output su video nel modo normale.
LA LINEA 130 stampa: PLUTO (in modo normale).
LA LINEA 140 fa terminare l'esecuzione del programma.

Formalismo sintattico per: APPLE II

Note:

— ESPRESSIONE NUMERICA può assumere i seguenti valori:
0 = caratteri in modo normale
1 = caratteri in modo "reverse"

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

NORMAL (Istruzione)

In APPLE predispone nel modo solito, cioè con lettere bianche su fondo nero, sia l'input che l'output del video.

Formalismo sintattico:

Esempio:

```
100 INVERSE 1
110 PRINT "PIPPO"
120 NORMAL
130 PRINT "PLUTO"
140 END
```

Risultato:

LA LINEA 100 predispone in modo inverso l'input e l'output su video.

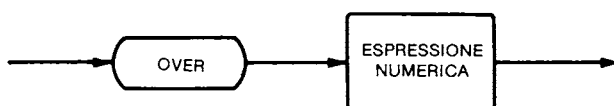
LA LINEA 110 stampa: **PIPPO** (in modo inverso).

LA LINEA 120 ripristina l'input e l'output su video in modo normale.
LA LINEA 130 stampa: PLUTO (in modo normale).

OVER (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Nello ZX Spectrum controlla la sovraimpressione per i caratteri stampati dopo la sua esecuzione. (0 = i caratteri stampati cancellano i caratteri precedentemente scritti, 1 = i nuovi caratteri sono stampati sopra quelli vecchi, e si ha il colore di primo piano dove uno dei due, ma non entrambi, hanno il colore di primo piano e il colore di sfondo dove entrambi hanno o il colore di sfondo o il colore di primo piano).

Formalismo sintattico:

Esempio:

```
10 OVER 1
20 PRINT AT 10, 10; "I"
30 PRINT AT 10, 10; "–"
40 OVER 0
```

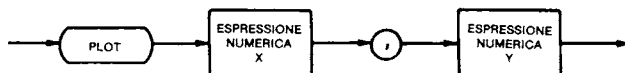
Risultato:

LINEA 10: definisce il modo di sovrapposizione dei caratteri.
LINEA 20: stampa una I alla riga 10 e colonna 10.
LINEA 30: stampa un trattino alla riga 10 e colonna 10, senza cancellare la I che era stata stampata in precedenza.
LINEA 40: ripristina il modo normale di stampa dei caratteri, in cui i caratteri stampati in una posizione cancellano i caratteri precedentemente presenti nella stessa posizione.

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

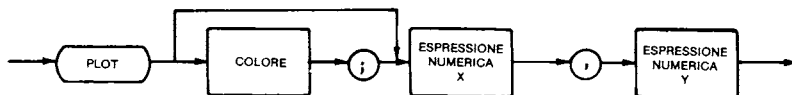
PLOT (Istruzione)

Nello ZX Spectrum e nello ZX 81 disegna un punto nella posizione specificata sullo schermo.
In Apple traccia un punto alle coordinate X e Y nel caso di grafica a bassa risoluzione, oppure evidenzia un carattere nel caso di funzionamento in modo testo.

Formalismo sintattico:

Esempio:

```
10 LET X = INT (RND * 256)
20 LET Y = INT (RND * 176)
30 PLOT X, Y
40 GO TO 10
```

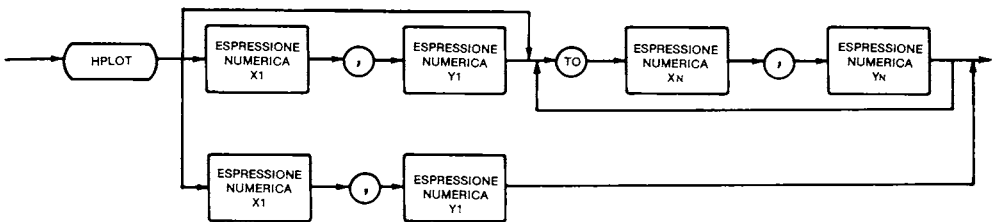
Risultato:
Il programma genera a caso dei valori per X e Y, che utilizza come coordinate dei puntini da accendere, e poi ricomincia da capo. In un certo tempo riempirà tutto il video.

Variante per: ZX Spectrum

Note:
— I valori possibili per ESPRESSIONE NUMERICA X ed ESPRESSIONE NUMERICA Y dipendono dal numero di punti in cui viene suddiviso il video, diverso tra i vari sistemi.
— COLORE è un'espressione numerica che modifica il colore di primo piano del punto disegnato.

HPLOT (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Può accendere un pixel alle coordinate X1, Y1, oppure può disegnare una linea dall'ultimo punto considerato fino a quello con coordinate X2, Y2, da qui ad un successivo punto di coordinate Xn, Yn, e così via.

Formalismo sintattico:

Esempio:

```
10 REM DISEGNA UN QUADRATO
20 HGR2
30 HPLOT 20, 20 TO 100, 20 TO 100, 100 TO 20, 100 TO 20, 20
```

Risultato:

LA LINEA 10 commenta il programma.
LA LINEA 20 predispone tutto il video in modo grafico ad alta risoluzione.
LA LINEA 30 disegna delle linee da un punto all'altro di quelli specificati, ed ottiene così un quadrato.

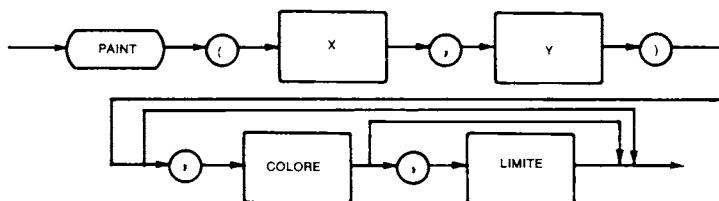
Note:

— Le X devono essere comprese tra 0 e 279, le Y tra 0 e 191.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

PAINT (Istruzione)

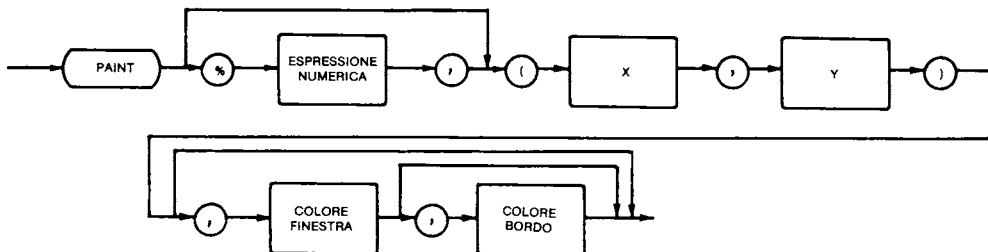
In M 20 colora l'intera finestra o una sua porzione, purchè delimitata da un contorno chiuso. In I.B.M. P.C. riempie un'area sullo schermo con il colore scelto.

Formalismo sintattico per: I.B.M. P.C.

Esempio:

```
10 SCREEN 1 : CLS
20 COLOR 2, 0
30 CIRCLE (75, 75), 50
40 PRINT (75, 75), 2, 3
```

Risultato:

LINEA 10: imposta il modo grafico a media risoluzione e pulisce lo schermo video.
LINEA 20: imposta il grigio per lo sfondo e sceglie la tavolozza 0.
LINEA 30: disegna un cerchio di raggio 50 con il centro alle coordinate (75, 75).
LINEA 40: colora l'interno del cerchio in rosso e il bordo del cerchio in marrone.

Formalismo sintattico per: M 20

Note per I.B.M. P.C.:

— X è un'espressione numerica che può assumere valori compresi tra 0 e 319.
— Y è un'espressione numerica che può assumere valori compresi tra 0 e 199.
— COLORE è un'espressione numerica che può assumere valori compresi tra 0 e 3. Definisce il colore dello sfondo dell'area da colorare.
— LIMITE è un'espressione numerica che può assumere valori compresi fra 0 e 3. Definisce il colore del bordo (perimetro) della figura da colorare.
— Per il significato dei codici di colore, vedere l'istruzione COLOR.

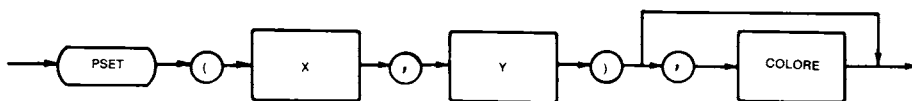
Note per Olivetti M20

— ESPRESSIONE NUMERICA può assumere valori compresi fra 1 e 16, e definisce la finestra su cui si sta lavorando.
— X è un'espressione numerica e può assumere valori compresi fra 0 e 511.
— Y è un'espressione numerica e può assumere valori compresi fra 0 e 255.
— COLORE FINESTRA è un'espressione numerica che può assumere valori compresi fra 0 e 3, e fa riferimento ai colori scelti nella specifica COLOR (1° Forma). Definisce il colore dello sfondo dell'area da colorare.
— COLORE BORDO è un'espressione numerica che può assumere valori compresi fra 0 e 3, e fa riferimento ai colori scelti nella specifica COLOR (1° Forma). Definisce il colore del bordo (perimetro) della figura da colorare.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

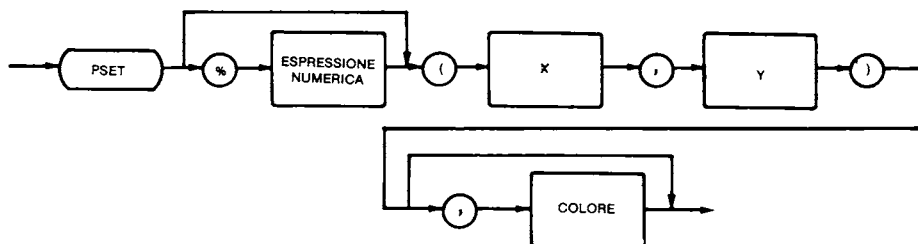
PSET (Istruzione)

In M 20 attiva con il colore specificato o con il colore di foreground il pixel più vicino al punto di coordinate (x, y) nella finestra specificata. In I.B.M. P.C. disegna un punto nella posizione specificata sullo schermo.

Formalismo sintattico per: I.B.M. P.C.

Esempio:

```
10 X = INT (RND * 320)
20 Y = INT (RND * 200)
30 PSET (X, Y), 1
40 GOTO 10
```

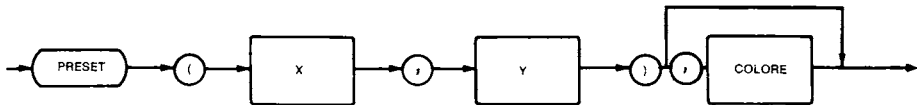
Risultato:
Il programma genera a caso dei valori per X e Y, che utilizza come coordinate dei punti da accendere, e poi ricomincia da capo. In un certo tempo riempirà tutto il video.

Formalismo sintattico per: M 20

Nota:
— per informazioni su ESPRESSIONE NUMERICA, X, Y e COLORE vedere, per esempio, le note all'istruzione PAINT.

PRESET (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

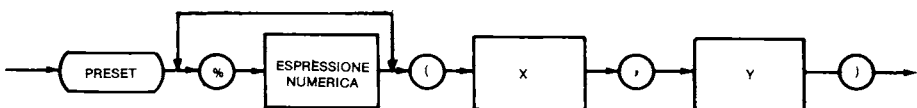
In M 20 attiva, con il colore di background della finestra specificata, il pixel più vicino al punto di coordinate (x, y). In I.B.M. P.C. disegna un punto nella posizione specificata sullo schermo.

Formalismo sintattico per: I.B.M. P.C.

Esempio:

```
10 X = INT (RND * 320)
20 Y = INT (RND * 200)
30 PSET (X, Y), 1
40 FOR I = I TO 10000 : NEXT I
50 PRESET (X, Y)
60 GOTO 10
```

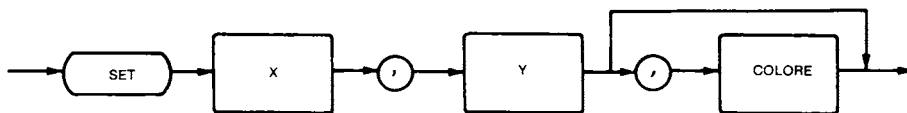
Risultato:

LINEA 10: genera un numero casuale compreso fra 0 e 319 e lo assegna alla variabile X.
LINEA 20: genera un numero casuale compreso fra 0 e 199 e lo assegna alla variabile Y.
LINEA 30: disegna un punto verde nella posizione determinata casualmente da X e Y.
LINEA 40: ciclo di ritardo per permettere la visualizzazione del punto colorato.
LINEA 50: cancella il punto (o lo disegna col colore di sfondo) alla stessa posizione in cui lo aveva disegnato in precedenza.
LINEA 60: riporta il controllo dell'esecuzione alla linea 10 per ricominciare da capo.

Formalismo sintattico per M 20

Nota:
— per informazioni su ESPRESSIONE NUMERICA, X, Y e COLORE vedere, per esempio, le note all'istruzione PAINT.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

SET (Istruzione)

Sul MZ 700 attiva i punti luminosi sulle coordinate del video specificate da X e Y.

Formalismo sintattico:

Esempio:

```
100 PRINT [C]
110 FOR I = 0 TO 49
120 SET I, I, 2
130 FOR K = 1 TO 1000 : NEXT K
140 RESET I, I
150 NEXT I
```

Risultato: sposta un punto luminoso in diagonale.
LINEA 100: pulisce il video.
LINEA 110: innesca un ciclo di 50 ripetizioni.
LINEA 120: attiva il punto luminoso alle coordinate (I, I), colorandolo di rosso.
LINEA 130: ciclo di ritardo per permettere la visualizzazione del punto.
LINEA 140: disattiva il punto luminoso precedentemente attivato.
LINEA 150: termine del ciclo innescato alla linea 110.

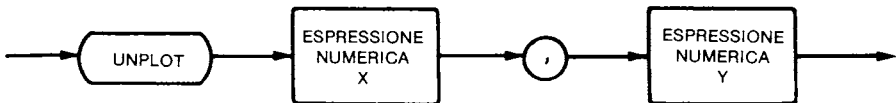
Note:
— X è un'espressione numerica con valori compresi tra 0 e 79.
— Y è un'espressione numerica con valori compresi tra 0 e 49.
— COLORE è un'espressione numerica con valori compresi tra 0 e 7 col seguente significato:

0 Nero
1 Blu
2 Rosso
3 Porpora
4 Verde
5 azzurro
6 Giallo
7 Bianco

UNPLOT (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Nello ZX 81 cancella il punto luminoso nella posizione specificata sullo schermo.

Formalismo sintattico:

Esempio:

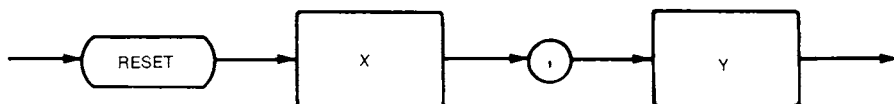
```
10 LET X = INT (RND * 64)
20 LET Y = INT (RND * 44)
30 PLOT X, Y
40 FOR I = 1 TO 1000
50 NEXT I
60 UN PLOT X, Y
70 GOTO 10
```

Risultato:

LINEA 10: genera un numero casuale compreso tra 0 e 63 e lo assegna alla variabile X.
LINEA 20: genera un numero casuale compreso tra 0 e 43 e lo assegna alla variabile Y.
LINEA 30: disegna un punto nella posizione determinata in modo casuale da X e Y.
LINEE 40÷50: ciclo di ritardo per permettere la visualizzazione del punto colorato.
LINEA 60: cancella il punto appena disegnato.
LINEA 70: riporta il controllo dell'esecuzione alla linea 10 per ricominciare da capo.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

RESET (Istruzione)

Sul MZ 700 disattiva i punti luminosi in corrispondenza delle coordinate dello schermo specificate da X e Y.

Formalismo sintattico:

Esempio:

```
100 PRINT [C]
110 FOR I = 0 TO 49
120 SET I, I, 2
130 FOR K = 1 TO 1000 : NEXT K
140 RESET I, I
150 NEXT I
```

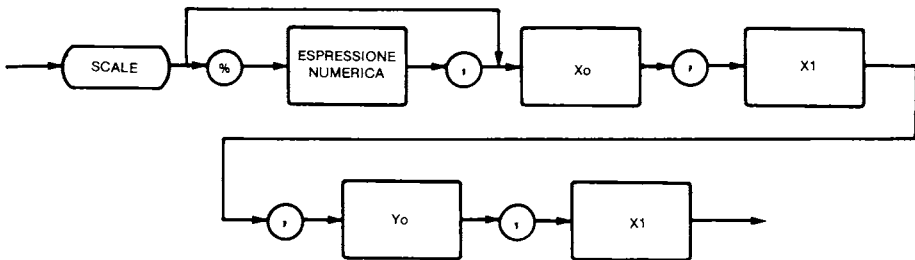
Risultato: sposta un punto luminoso in diagonale.
LINEA 100: pulisce il video.
LINEA 110: innesca un ciclo di 50 ripetizioni.
LINEA 120: attiva il punto luminoso alle coordinate (I,I), colorandolo di rosso.
LINEA 130: ciclo di ritardo per permettere la visualizzazione del punto.
LINEA 140: disattiva il punto luminoso precedentemente attivato.
LINEA 150: termine del ciclo innescato alla linea 110.

Note:
— X è un'espressione numerica con valori compresi tra 0 e 79.
— Y è un'espressione numerica con valori compresi tra 0 e 49.

SCALE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

Definisce la trasformazione tra il sistema di coordinate scelto dall'utente e il sistema di coordinate standard.

Formalismo sintattico:

Esempio:

```
10 CIRCLE (100, 100), 50
20 SCALE - 1000, 1000, 0, 2000
30 CIRCLE (100, 100), 50
```

Risultato:

LINEA 10: disegna un cerchio di raggio 50, con centro alle coordinate (100,100).
LINEA 20: trasforma il sistema di coordinate standard.
LINEA 30: disegna un cerchio di raggio 50 con centro alle coordinate (100,100) nel nuovo sistema di coordinate: tale cerchio sarà nettamente diverso da quello disegnato in precedenza.

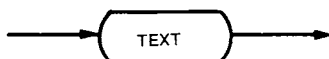
Note:

— ESPRESSIONE NUMERICA può assumere valori compresi tra 1 e 16 e individua la finestra su cui si stà lavorando.
— X0 e X1 sono espressioni numeriche che identificano, rispettivamente, le ascisse del lato sinistro e destro della finestra.
— Y0 e Y1 sono espressioni numeriche che identificano, rispettivamente, le ordinate del lato inferiore e superiore della finestra.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

TEXT (Istruzione)

Definisce la visualizzazione nel modo testo a schermo intero (24 righe da 40 caratteri ciascuna). Il carattere di richiesta (PROMPT) e il cursore sono spostati all'ultima riga dello schermo.

Formalismo sintattico:

Esempio:

```
10 GR
20 COLOR = 5
30 HLIN 10, 20 AT 1
40 TEXT
50 END
```

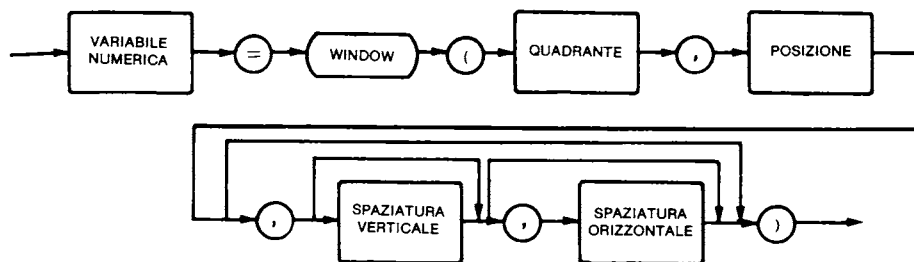
Risultato:

LA LINEA 10 cancella il video e lo predispone in modo grafico.
LA LINEA 20 definisce il colore.
LA LINEA 30 disegna una linea orizzontale dalla colonna 10 alla colonna 20 sulla riga 1.
LA LINEA 40 ridefinisce il video nel modo testo a schermo intero.

WINDOW (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

Suddivide la finestra attuale, quella cioè su cui si sta operando, permettendo l'apertura di una nuova finestra, a cui assegna automaticamente un numero d'ordine.

Formalismo sintattico:

Esempio:

```
10 A = WINDOW (0, 90)
20 WINDOW % 2
30 CIRCLE (10, 10), 5
40 WINDOW % 1
```

Risultato:

LINEA 10: apre una nuova finestra a cui il sistema assegnerà automaticamente il numero d'ordine 2.

LINEA 20: seleziona la finestra appena generata, che diventa la finestra attuale.

LINEA 30: disegna, all'interno della finestra selezionata, un cerchio di raggio 5 con centro alle coordinate (10,10).

LINEA 40: seleziona la finestra 1, cioè l'intero video, come finestra attuale.

Note:

— QUADRANTE è un'espressione numerica che specifica la posizione, rispetto alla finestra genitrice, in cui viene aperta la nuova finestra. Può assumere i seguenti valori:

0 nella parte alta
1 nella parte bassa
2 nella parte sinistra
3 nella parte destra

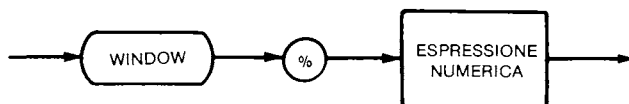
— POSIZIONE è una espressione numerica il cui valore dipende dalla opzione QUADRANTE.

— SPAZIATURA VERTICALE è un'espressione numerica che può assumere valori compresi tra 10 e 16.

— SPAZIATURA ORIZZONTALE è un'espressione numerica che può assumere i due valori 6 o 8.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

WINDOW (Istruzione)

Seleziona la finestra su cui operare, che diviene la finestra attuale.

Formalismo sintattico:

Esempio:

```
10 A = WINDOW (0, 90)
20 WINDOW % 2
30 CIRCLE (10, 10), 5
40 WINDOW % 1
```

Risultato:

LINEA 10: Apre una nuova finestra a cui il sistema assegnerà automaticamente il numero d'ordine 2.
LINEA 20: Seleziona la finestra appena generata, che diventa la finestra attuale.
LINEA 30: Disegna, all'interno della finestra selezionata, un cerchio di raggio 5 con centro alle coordinate (10,10).
LINEA 40: Seleziona la finestra 1, cioè l'intero video, come finestra attuale.

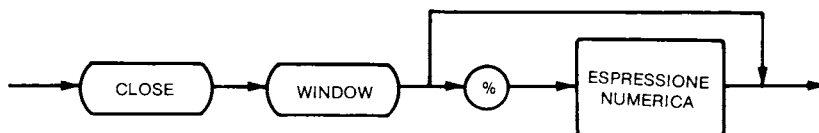
Nota:

— ESPRESSIONE NUMERICA può assumere valori compresi tra 1 e 16.

CLOSE WINDOW (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

Chiude una finestra specificata oppure tutte le finestre aperte.

Formalismo sintattico:

Esempio:

```
10 A = WINDOW (0, 90)
20 WINDOW % 2
30 CIRCLE (10, 10), 5
40 WINDOW % 1
50 CLOSE WINDOW % 2
```

Risultato:

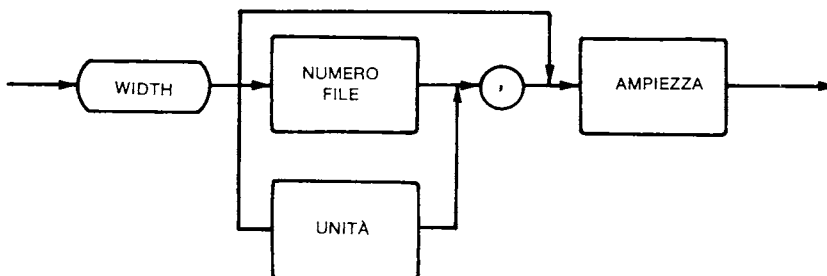
LINEA 10: Apre una nuova finestra a cui il sistema assegnerà automaticamente il numero d'ordine 2.
LINEA 20: Seleziona la finestra appena generata, che diventa la finestra attuale.
LINEA 30: Disegna, all'interno della finestra selezionata, un cerchio di raggio 5 con centro alle coordinate (10,10).
LINEA 40: Seleziona la finestra 1, cioè l'intero video, come finestra attuale.
LINEA 50: Chiude la finestra aperta in precedenza alla linea 10.

Note:
— ESPRESSIONE NUMERICA può assumere valori compresi tra 1 e 16.
— la finestra principale (n. d'ordine 1) non può mai venir chiusa.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

WIDTH (Comando - istruzione)

In I.B.M. P.C. definisce l'ampiezza della riga di emissione in numeri di caratteri. Dopo aver emesso il numero caratteri indicato, il BASIC aggiunge un ritorno a margine.

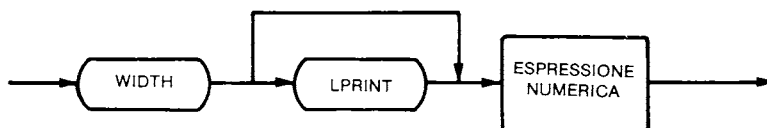
Formalismo sintattico per: I.B.M. P.C.

Esempio:

```
 10 WIDTH 80
 20 WIDTH "LPT1:", 132
 30 OPEN "LPT1:" FOR OUTPUT AS # 1
    :
900 WIDTH # 1, 80
```

Risultato:

LINEA 10: Imposta l'ampiezza dello schermo in 80 colonne.
LINEA 20: Imposta l'ampiezza per la stampante 1 di 132 caratteri.
LINEA 30: Apre il file di uscita alla stampante. All'esecuzione di questa istruzione viene effettivamente modificata l'ampiezza della stampante.
LINEA 900: Cambia l'ampiezza della stampante, trasformandola in 80 caratteri per riga.

In M 20 definisce l'ampiezza massima della linea del video o della stampante.

Formalismo sintattico per: M 20

Note:
— NUMERO FILE è composto dal simbolo # seguito da un'espressione numerica che può assumere valori compresi tra 1 e 15.
— UNITA' è un'espressione a stringa che identifica l'unità periferica. Vedere l'allegato A per l'elenco delle unità ammesse.
— AMPIEZZA è un'espressione numerica che può assumere valori compresi tra 0 e 255. Per lo schermo è possibile utilizzare solo 40 o 80.

ATTR (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Nello ZX Spectrum ritorna gli attributi di un dato carattere dello schermo, alla riga X e colonna Y. Il valore ritornato è il risultato della somma di 4 numeri, secondo lo schema seguente:
- 128 se la posizione è lampeggiante, 0 se è stabile.
- 64 se la posizione è extra luminosa, 0 se è normale.
- 8 volte il codice del colore di fondo.
- Il codice del colore di primo piano.

Formalismo sintattico:

Esempio:

```
10 FLASH 1
20 PAPER 7
30 INK 0
40 PRINT AT 10, 10; "A"
50 PRINT AT 0, 0; ATTR (10, 10)
```

Risultato:

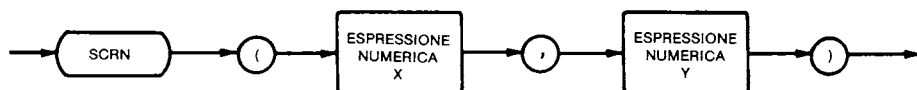
LINEA 10: imposta il lampeggiamento dei caratteri.
LINEA 20: imposta il bianco come colore di fondo.
LINEA 30: imposta il nero come colore di primo piano.
LINEA 40: stampa il carattere A alle coordinate (10,10).
LINEA 50: stampa il valore degli attributi del carattere A stampato in precedenza alle coordinate (10,10), in un'altra posizione del video. Si otterrà:
184
(che è dato da 128 + 8 * 7)

Note:
— ESPRESSIONE NUMERICA X può assumere valori compresi fra 0 e 31.
— ESPRESSIONE NUMERICA Y può assumere valori compresi fra 0 e 23.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

SCRN (Funzione)

Nel modo grafico a bassa risoluzione, la funzione SCRN ritorna il codice del colore del punto alle coordinate X, Y.

Formalismo sintattico:

Esempio:

```
10 HGR
20 HCOLOR = 5
30 DRAW 1 AT 100, 70
40 PRINT SCRN (100, 70)
50 END
```

Risultato:

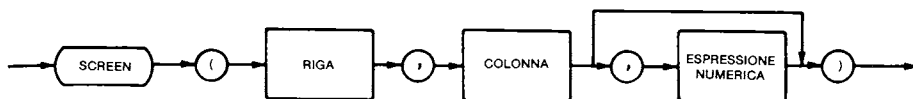
LA LINEA 10 cancella il video e lo predispone in alta risoluzione, modo grafico.
LA LINEA 20 definisce il colore usato.
LA LINEA 30 definisce l'inizio del profilo 1 all'intersezione delle coordinate 100 e 70, e lo disegna.
LA LINEA 40 ritorna il codice di colore del punto (100,70), che è il valore assegnato alla linea 20. Viene dunque stampato: 5.

Note:
— ESPRESSIONE NUMERICA X può assumere valori compresi tra 0 e 39.
— ESPRESSIONE NUMERICA Y può assumere valori compresi tra 0 e 47.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

SCREEN (Funzione)

IN I.B.M. P.C. ritorna il codice ASCII ($0 \div 255$) per il carattere sullo schermo attivo alla riga e colonna specificata, oppure l'attributo del colore del carattere stesso.

Formalismo sintattico:

Esempio:

```
 300 A = SCREEN (1, 2)
 310 IF CHR$ (A) = "P" THEN 2000
      :
2000 REM *
```

Risultato:

LA LINEA 300 assegna alla variabile A il codice ASCII del carattere in riga 1 colonna 2.

LA LINEA 310, se il carattere è P, salta alla 2000.

Note:

— RIGA è un'espressione numerica che può assumere valori compresi fra 1 e 25.

— COLONNA è un'espressione numerica che può assumere valori compresi tra 1 e 40, oppure tra 1 e 80, in dipendenza del valore impostato nella WIDTH.

— ESPRESSIONE NUMERICA è valida solo in modo testo, e può essere solo falsa (zero) o vera (non-zero). Se specificata, fa in modo che la funzione SCREEN ritorni l'attributo del colore del carattere al posto del codice ASCII del carattere.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

SCREEN$ (Funzione)

Nello ZX Spectrum ritorna il carattere che appare sul video, sia in normale che in campo inverso, alla riga X e colonna Y.

Formalismo sintattico:

Esempio:

```
10 FLASH 1
20 INVERSE 1
30 PRINT AT 10, 10; "PIPPO"
40 PRINT AT (0, 0); SCREEN$ (14, 10)
```

Risultato:

LINEA 10: imposta il lampeggiamento dei caratteri.

LINEA 20: imposta il modo "reverse" dei caratteri (scambio del colore di primo piano con quello dello sfondo e viceversa).

LINEA 30: stampa la parola specificata iniziando al punto di coordinate (10,10).

LINEA 40: stampa, alle coordinate (0,0), il carattere che intercetta sul video alle coordinate (14, 10), e cioè la "O" di PIPPO. Questo anche se il carattere è lampeggiante e in modo "reverse".

Note:

— ESPRESSIONE NUMERICA X può assumere valori compresi fra 0 e 31.

— ESPRESSIONE NUMERICA Y può assumere valori compresi fra 0 e 23.

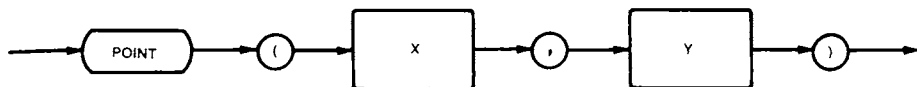
	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

POINT (Funzione)

Per M 20 assegna ad una variabile il numero di colore del pixel più vicino alle coordinate (X, Y) specificate, all'interno della finestra attuale.

Per I.B.M. P.C. ritorna il colore del punto specificato sullo schermo.

Nello ZX Spectrum viene ritornato 1 se il pixel ha il colore di primo piano, 0 se ha il colore dello sfondo.

Formalismo sintattico:

Esempio:

```
10 SCREEN 1 : CLS
20 FOR X = 0 TO 319
30 FOR Y = 0 TO 199
40 IF POINT (X, Y) = 0 THEN PSET (X, Y), 2
50 IF POINT (X, Y) < > 0 THEN PRESET (X, Y)
60 NEXT Y
70 NEXT X
```

Risultato:

LINEA 10: imposta il modo grafico a media risoluzione e pulisce lo schermo video.

LINEA 20: inizia un ciclo di 320 ripetizioni.

LINEA 30: inizia un ciclo di 200 ripetizioni interno al precedente. I due cicli scorrono in pratica l'intero video.

LINEA 40: se il colore del punto alle coordinate (X,Y) è uguale a zero, cioè al colore di fondo, viene impostato un colore rosso.

LINEA 50: se il colore del punto alle coordinate (X,Y) non è uguale al colore di fondo, il punto viene spento. Il risultato dell'esecuzione delle linee 40 e 50 è che tutti i punti spenti del video vengono colorati in rosso e tutti i punti accesi vengono spenti.

LINEE 60÷70: termine dei cicli iniziati, rispettivamente, alla linea 30 e alla linea 20.

Note:

— X è un'espressione numerica che può assumere valori compresi fra 0 e 319 per I.B.M. P.C., tra 0 e 511 per Olivetti M20 e tra 0 e 255 per ZX Spectrum.

— Y è un'espressione numerica che può assumere valori compresi fra 0 e 199 per I.B.M. P.C., tra 0 e 255 per Olivetti M20 e tra 0 e 191 per ZX Spectrum.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

SCALEX (Funzione)

Trasforma una coordinata utente nella corrispondente coordinata standard, sull'asse delle X.

Formalismo sintattico:

Esempio:

```
10 SCALE - 1000, 1000, 0, 2000
20 CIRCLE (100, 150), 50
30 X = SCALE X (100)
40 Y = SCALE Y (150)
50 SCALE 0, 511, 0, 255
60 CIRCLE (X, Y), 50
```

Risultato:

LINEA 10: trasforma il sistema di coordinate con valori definiti dall'utente.

LINEA 20: disegna un cerchio di raggio 50 unità, con centro alle coordinate (100, 150) del nuovo sistema di coordinate.

LINEA 30: trasforma la coordinata utente orizzontale 100 nella corrispondente coordinata standard, e memorizza il risultato nella variabile X.

LINEA 40: trasforma la coordinata utente verticale 150 nella corrispondente coordinata standard, e memorizza il risultato nella variabile Y.

LINEA 50: ripristina il sistema di coordinate standard.

LINEA 60: disegna un cerchio di raggio 50 unità standard, con centro nello stesso punto in cui era previsto il centro del cerchio, disegnato alla linea 20, nelle coordinate utente.

SCALEY (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

Trasforma una coordinata utente nella corrispondente coordinata standard sull'asse delle Y.

Formalismo sintattico:

Esempio:

```
10 SCALE - 1000, 1000, 0, 2000
20 CIRCLE (100, 150), 50
30 X = SCALE X (100)
40 Y = SCALE Y (150)
50 SCALE 0, 511, 0, 255
60 CIRCLE (X, Y), 50
```

Risultato:

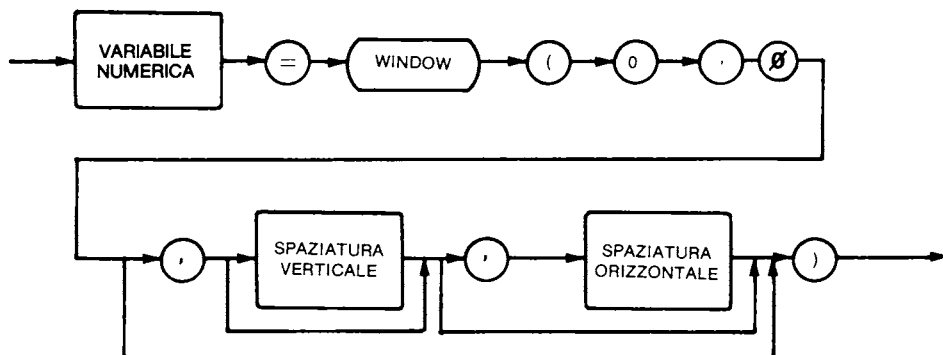
LINEA 10: trasforma il sistema di coordinate con valori definiti dall'utente.
LINEA 20: disegna un cerchio di raggio 50 unità, con centro alle coordinate (100, 150) del nuovo sistema di coordinate.
LINEA 30: trasforma la coordinata utente orizzontale 100 nella corrispondente coordinata standard, e memorizza il risultato nella variabile X.
LINEA 40: trasforma la coordinata utente verticale 150 nella corrispondente coordinata standard, e memorizza il risultato nella variabile Y.
LINEA 50: ripristina il sistema di coordinate standard.
LINEA 60: disegna un cerchio di raggio 50 unità standard, con centro nello stesso punto in cui era previsto il centro del cerchio, disegnato alla linea 20, nelle coordinate utente.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

WINDOW (Funzione)

Con questa funzione non viene aperta alcuna nuova finestra.

Essa viene usata per variare lo spazio fra i caratteri e/o lo spazio fra linee di testo nella finestra attuale.

Formalismo sintattico:

Esempio:

```
10 A = WINDOW (0, 90, 10, 6)
20 WINDOW % A
30 CIRCLE (10, 10), 5
40 A = WINDOW (0, 0, 16, 8)
50 CIRCLE (10, 10), 5
60 WINDOW % 1
```

Risultato:

LINEA 10: apre una nuova finestra a cui il sistema assegnerà automaticamente il numero d'ordine 2.

LINEA 20: seleziona la finestra appena generata, che diventa la finestra attuale.

LINEA 30: disegna un cerchio di raggio 5 unità, con centro alle coordinate (10,10), all'interno della finestra specificata.

LINEA 40: modifica la spaziatura verticale a 16 e quella orizzontale a 8 relativamente alla finestra 2.

LINEA 50: disegna un cerchio di raggio 5 unità, con centro alle coordinate (10,10). Poichè le spaziature sono cambiate, questo cerchio non sarà uguale a quello disegnato alla linea 30.

LINEA 60: seleziona la finestra 1, cioè l'intero video, come finestra attuale.

Note:
— per SPAZIATURA VERTICALE e SPAZIATURA ORIZZONTALE, vedere l'istruzione WINDOW per generare una finestra.

GENERAZIONE CARATTERI

Generalità

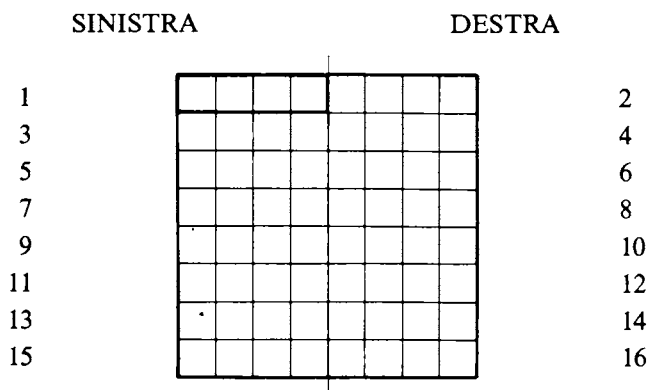
Ogni carattere è normalmente costituito da una matrice di 8 x 8 punti luminosi (pixel).

I caratteri vengono creati illuminando alcuni dei 64 pixel e lasciandone spenti altri.

I caratteri standard risiedono normalmente in ROM, ad indirizzi opportuni, e vengono richiamati all'occorrenza da un opportuno elemento dell'elaboratore. In alcuni calcolatori è possibile che l'utente definisca da sé dei particolari caratteri inesistenti nella configurazione standard. Le modalità di definizione e di memorizzazione dei caratteri generati dall'utente, dipendono dal tipo di elaboratore utilizzato.

TEXAS TI 99/4A

I 64 pixel che formano la matrice del carattere vengono suddivisi in 16 gruppi di 4. Ad ogni pixel spento viene fatto corrispondere il valore logico 0, ad ogni pixel acceso il valore logico 1.

SINISTRA	DESTRA
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16

Con quattro elementi binari è possibile rappresentare $2^4 = 16$ oggetti diversi, per cui ogni gruppo di 4 pixel può essere rappresentato da un numero esadecimale.

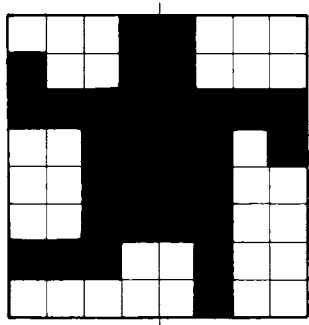
L'insieme dei sedici numeri esadecimali forma una stringa da inserire nell'istruzione CALL CHAR.

Esempio:

	BINARIO		ESADECIMALE	
	0001	1000	1	8
	1001	1000	9	8
	1111	1111	F	F
	0011	1101	3	D
	0011	1100	3	C
	0011	1100	3	C
	1110	0100	E	4
	0000	0100	0	4

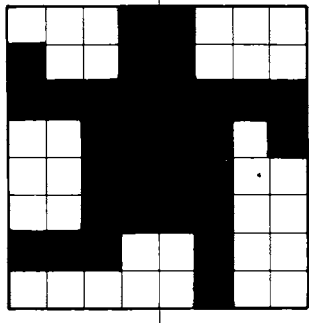
Per descrivere la sagoma dell'esempio si dovrà inserire quindi, nell'istruzione CALL CHAR, la seguente espressione stringa:

"1898FF3D3C3CE404"

Tali caratteri così generati, verranno memorizzati, a scelta, in sovrapposizione a caratteri già presenti, oppure in nuove aree di memoria. Ciò dipende dal codice ASCII del carattere che si utilizza nell'istruzione CALL CHAR: se il codice sarà compreso fra 32 e 127, andrà a sostituire l'equivalente carattere in memoria, che non sarà quindi più disponibile per quel programma; se il codice sarà compreso fra 128 e 159 verrà generata una nuova configurazione. I caratteri generati dall'utente valgono finchè il programma che li genera resta in esecuzione: al termine viene ripristinata la configurazione standard.

SINCLAIR ZX SPECTRUM

È possibile definire fino a 21 caratteri grafici diversi contemporaneamente, utilizzando i codici ASCII da 144 a 164 (A÷U). La scelta del codice da utilizzare viene effettuata con la funzione USR. Ad esempio, la funzione USR "P" memorizzerà i primi 8 pixel, cioè il primo gruppo di 8 bit (prima riga) della matrice, la funzione USR "P" +1 memorizzerà il secondo gruppo, e così via fino alla USR "P" +7 che memorizzerà l'ultima riga. In tal modo si è memorizzato l'intero carattere studiato nel carattere che corrisponde al codice ASCII 159 (la P).

Per inserire la configurazione di pixel accesi e spenti che determinano la sagoma voluta, si fa uso dell'istruzione BIN.

In tale istruzione, che è valida per un solo gruppo di 8 pixel, si metteranno 0 o 1 in dipendenza dei pixel spenti o accesi.

Esempio:

	BIN	
	BIN	00011000
	BIN	10011000
	BIN	11111111
	BIN	00111101
	BIN	00111100
	BIN	00111100
	BIN	11100100
	BIN	00000100

COMMODORE CBM 64

Le configurazioni dei caratteri standard sono memorizzate nella ROM. Tuttavia la locazione di memoria che indirizza a tali caratteri è programmabile, per cui è possibile modificarla per indirizzare alla RAM. È però opportuno, prima di inserire i caratteri programmabili dell'utente, copiare da ROM a RAM tutti i caratteri standard.· Dopo questa operazione è possibile inserire le configurazioni volute, utilizzando l'istruzione POKE ed eventualmente la funzione PEEK, ricordando che ogni carattere è dato da una matrice 8 x 8 e che la POKE può indirizzare un byte (8 bit) alla volta: per memorizzare un carattere sono quindi necessarie 8 POKE a indirizzi contigui.

Per esempio, si supponga di voler memorizzare, nel carattere 63, e dopo aver copiato tutti gli altri, la seguente configurazione:

BINARIO	DECIMALE
00011000	24
10011000	152
11111111	255
00111101	61
00111100	60
00111100	60
11100100	228
00000100	4

Le operazioni da effettuarsi sono le seguenti:

```
DISATTIVARE LE INTERRUZIONI ED ESCLUDERE L'I/O:
10 POKE 56334, PEEK (56334) AND 254
20 POKE 1, PEEK (1) AND 251
COPIARE I PRIMI 63 CARATTERI DA ROM A RAM, PARTENDO DALL'INDIRIZZO
12288 (PER ESEMPIO):
30 FOR I = 0 TO 62
40 FOR J = 0 TO 7
50 POKE 12288 + I * 8 + J, PEEK (12288 + I * 8 + J)
60 NEXT J
70 NEXT I
RIATTIVARE LE INTERRUZIONI E L'I/O:
80 POKE 1, PEEK (1) OR 4
90 POKE 56334, PEEK (56334) OR 1
IMPOSTARE IL PUNTATORE DEI CARATTERI VERSO LA LOCAZIONE RAM
12288:
100 POKE 53272, (PEEK (53272) AND 240) + 12
PROGRAMMARE IL CARATTERE 63 DEFINITO PRIMA:
110 CHAR = 63
120 FOR BYTE = 0 TO 7
130 READ NUMBER
140 POKE 12288 + (8 * CHAR) + BYTE, NUMBER
150 NEXT BYTE
UTILIZZARE IL CARATTERE DEFINITO ORA (RESTO DEL PROGRAMMA):
INSERIRE LE ISTRUZIONI DATA CHE RACCHIUDONO LE CARATTERISTI-
CHE DELLA SAGOMA:
9000 DATA 24, 152, 255, 61, 60, 60, 228, 4
```

COMMODORE VIC 20

Anche in questo elaboratore, come per il fratello maggiore, il CBM 64, le configurazioni dei caratteri standard sono memorizzate nella ROM, ma possono essere spostate nella RAM, ad indirizzi voluti. Modificando poi il puntatore alla mappa caratteri, è possibile prelevare le informazioni-carattere dalla RAM. A questo punto è anche possibile modificare, con delle POKE, le configurazioni dei caratteri per inserire nuove configurazioni definite dall'utente. Il problema che si pone nel caso del VIC 20 senza espansione di memoria, quindi con memoria utente di circa 3,5 Kbytes, è che, ricopiando dalla ROM il set di 256 caratteri, questo occuperebbe 2 Kbytes, lasciandone all'utente solo 1536. Si può ovviare a questo inconveniente non ricopiando l'intero set di caratteri ma solo quelli effettivamente utilizzati o, meglio ancora, non copiando niente ma modificando temporaneamente il puntatore alla mappa carattere solo quando si utilizzano effettivamente i

caratteri definiti dall'utente, e ripristinando subito dopo il puntatore al suo valore abituale.

Giova ricordare che, contrariamente a quanto accade nel TI 99/4A, la mappa caratteri RAM, una volta riempita, resta inalterata fin quando si spegne il calcolatore, per cui può essere utilizzata da programmi diversi.

Segue un esempio analogo a quello visto per il CBM 64, che ricopia 128 caratteri da ROM a RAM e modifica il carattere 63 con la configurazione "omino".

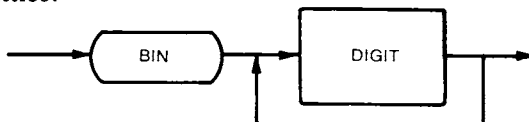
Le operazioni da effettuare sono le seguenti:

```
RISERVARE SPAZIO PROTETTO IN MEMORIA:
10 POKE 52,24
20 POKE 56,24
30 CLR
COPIARE I PRIMI 128 CARATTERI DA ROM A RAM, PARTENDO DALLA
LOCAZIONE 5120
40 FOR J = 0 TO 1023
50 POKE 5120 + J, PEEK (32768 + J)
60 NEXT J
PROGRAMMARE IL CARATTERE 63 DEFINITO IN PRECEDENZA:
70 CHAR = 63
80 FOR BYTE = 0 TO 7
90 READ NUMBER
100 POKE 5120 + (8 * CHAR) + BYTE, NUMBER
110 NEXT BYTE
IMPOSTARE IL PUNTATORE DI CARATTERI VERSO LA LOCAZIONE
5120 :
120 POKE 36869, 253
UTILIZZARE IL CARATTERE ORA DEFINITO (RESTO DEL PROGRAMMA)
INSERIRE LE ISTRUZIONI DATA CONTENENTI LE CARATTERISTICHE DEL-
LA SAGOMA:
9000 DATA 24, 152, 255, 61, 60, 60, 228, 4
```

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

BIN (Istruzione)

Consente di definire, nello ZX Spectrum, caratteri speciali non presenti nella configurazione standard.

Formalismo sintattico:

Esempio:

```
   10 FOR I = 0 TO 7
   20 READ B
   30 POKE USR "P" + I, B
   40 NEXT I
   50 :
8990
9000 DATA  BIN  00011000
9010 DATA  BIN  10011000
9020 DATA  BIN  11111111
9030 DATA  BIN  00111101
9040 DATA  BIN  00111100
9050 DATA  BIN  00111100
9060 DATA  BIN  11100100
9070 DATA  BIN  00000100
```

Risultato: genera il carattere "omino" e lo memorizza al posto del carattere P, codice ASCII 159 (vedere paragrafo "generazione caratteri").

LINEA 10: inizia un ciclo di otto ripetizioni.

LINEA 20: legge una istruzione DATA e la memorizza nella variabile numerica B.

LINEA 30: modifica gli indirizzi di memoria che contengono il carattere P con codice ASCII 159, dall'indirizzo USR "P" all'indirizzo USR "P" +7, con i valori binari che contengono la configurazione "omino".

LINEA 40: fine del ciclo iniziato alla linea 10.

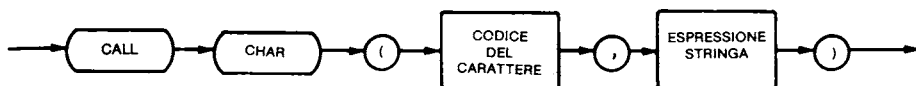
LINEE 50÷8990: resto del programma, compreso l'utilizzo del carattere "omino" ora generato.

LINEE 9000÷9070: elenco delle istruzioni DATA che contengono le istruzioni BIN con la configurazione del carattere "omino".

CALL CHAR (Istruzione)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Consente di definire, nel TEXAS TI 99/4A, speciali caratteri grafici non presenti nella configurazione standard.

Formalismo sintattico:

Esempio:

```
100 CALL CLEAR
110 CALL CHAR (65, "1898FF3D3C3CE404")
120 CALL COLOR (5, 2, 14)
130 CALL HCHAR (10, 15, 65, 4)
140 END
```

Risultato:

LINEA 100: pulisce il video.
LINEA 110: genera il carattere "omino" (vedere il paragrafo "generazione caratteri") e lo memorizza, con codice ASCII 65, al posto della lettera A.
LINEA 120: specifica che il quinto gruppo di caratteri (quelli con codice ASCII da 64 a 71) verrà stampato in nero su sfondo magenta.
LINEA 130: stampa a video il carattere "omino" alla riga 10 e colonna 15, e lo ripete quattro volte in orizzontale.
LINEA 140: fa terminare l'esecuzione del programma. I quattro omini presenti sul video ritorneranno delle lettere A con i colori standard.

Nota:

— CODICE DEL CARATTERE è un'espressione numerica che può assumere i valori da 32 a 159. Quando avrà valore compreso fra 32 e 127, il nuovo carattere prenderà temporaneamente il posto di quello vecchio; se il valore sarà compreso tra 128 e 159 verrà generato un nuovo carattere, sempre che vi sia abbastanza spazio in memoria.
— Alla fine del programma tutti i caratteri compresi tra i codici ASCII 32 e 127 ritornano alla loro normale rappresentazione, mentre quelli con codice ASCII da 128 a 159 rimangono non definiti.
— Questa istruzione serve solo a definire il carattere: per far uso delle istruzioni di stampa (CALL HCHAR, CALL VCHAR, ecc.).

GENERAZIONE PROFILI

Generalità

Abbiamo visto che, in tutti gli elaboratori trattati, un carattere è definito come una matrice di 8 x 8 (cioè 64) punti luminosi o pixel. In alcuni di questi elaboratori (TI 99/4A, ZX Spectrum, VIC 20, CBM 64) è possibile per l'utente generare delle configurazioni particolari di queste matrici, cioè creare dei caratteri a sua scelta. Questo è valido in modo testo, non in modo grafico. Nell'I.B.M. P.C., nell'Olivetti M 20 e nell'APPLE II non è invece possibile generare configurazioni di caratteri (in modo testo), richiamabili mediante il loro codice ASCII, diverse da quelle previste dal costruttore. È però possibile, in modo grafico, generare delle configurazioni qualsiasi di pixel (che chiameremo "profili") e movimentarle a piacere sul video. I profili differiscono dai caratteri sia perchè la loro grandezza non è limitata alla matrice 8 x 8, ma può estendersi per un numero voluto di pixel, sia perchè non hanno un codice ASCII che le contraddistingua.

Le configurazioni possibili di profili possono variare da figure geometriche create con le istruzioni CIRCLE e LINE (cerchi, ellissi, quadrati, rettangoli, ecc.) a figure più complesse, disegni per giochi, riproduzioni di opere d'arte e così via. È possibile far muovere i profili ovunque sullo schermo video: il movimento è in tal caso più omogeneo, più fluido che non il movimento creato da caratteri sullo schermo.

I.B.M. P.C.

Per generare un profilo bisogna innanzitutto entrare nel modo grafico, indi disegnare sul video la figura voluta, per punti o per linee, utilizzando le normali istruzioni grafiche (LINE, CIRCLE, DRAW, ecc.). La figura può essere disegnata in qualsiasi posizione del video.

Per poterla poi animare, è necessario che la figura stessa venga memorizzata in una matrice, utilizzando l'istruzione GET. In questa istruzione si devono dichiarare le coordinate del punto più in alto a sinistra e le coordinate del punto più in basso a destra della figura da memorizzare (come se fosse contenuta quindi nel rettangolo definito dai due punti).

Con l'istruzione PUT è poi possibile riproporre la figura in qualsiasi momento e in qualsiasi punto del video. Le dimensioni della matrice che deve contenere il profilo possono venir calcolate con la seguente formula:

4 + INT ((X * HMR + 7) /8) * Y

dove:

X = lunghezza del lato orizzontale del rettangolo che contiene il profilo
Y = lunghezza del lato verticale del rettangolo che contiene il profilo
HMR = 2 se media risoluzione
HMR = 1 se alta risoluzione

Come esempio di generazione e di utilizzo di profili proponiamo un gioco completo: un maiale, chiuso in una gabbia, lancia frecce contro degli omini che scendono dal cielo attaccati a palloncini. Se le frecce colpiscono i palloncini, gli omini cadono velocemente a testa in giù. La gabbia si può muovere verticalmente utilizzando i tasti "8" e "2", mentre la barra spaziatrice serve per far partire le frecce.

Spieghiamo brevemente i punti essenziali del programma.

LINEA 40: dimensiona la matrice BALL% per l'omino col palloncino.

LINEA 50: seleziona la grafica a media risoluzione, pulisce il video, determina il colore di primo piano e di sfondo. Queste tre istruzioni sono obbligatorie nella generazione dei profili.

LINEE 60÷110: istruzioni grafiche per disegnare l'omino col palloncino.

LINEA 120: memorizza il profilo dell'omino col palloncino nella matrice BALL%.

LINEA 130: pulisce il video.

LINEA 140: dimensiona la matrice MAN% per l'omino a testa in giù.

LINEE 150÷160: istruzioni grafiche per disegnare l'omino a testa in giù.

LINEA 170: memorizza il profilo dell'omino a testa in giù nella matrice MAN%.

LINEA 200: memorizza un profilo completamente a spazi (per operazioni di cancellazione di profili precedenti).

LINEE 220÷490: istruzioni grafiche per disegnare il maialino in gabbia.

LINEA 500: dimensiona la matrice PIG%.

LINEA 510: memorizza il profilo del maialino in gabbia nella matrice PIG%.

LINEE 550÷560: istruzioni grafiche per disegnare la freccia.

LINEA 570: dimensiona la matrice DARK%.

LINEA 580: memorizza il profilo della freccia nella matrice DARK%.

LINEA 720: esplora la tastiera per controllare se è stato digitato un carattere, che memorizza nella variabile A$.

LINEA 810: visualizza il profilo dell'omino col palloncino. La coordinata delle X (variabile B1) è scelta in modo casuale alla linea 630.

LINEA 820: eventualmente fa partire una freccia.

LINEE 830÷840: aggiungono o tolgono una costante alla variabile D1 che determina la coordinata verticale del maialino in gabbia. Servono quindi per spostare la gabbia verso l'alto o verso il basso.

LINEA 870: visualizza il profilo del maialino in gabbia.

LINEA 900: toglie una costante alla variabile L1 che determina la coordinata orizzontale della freccia e visualizza quindi la freccia leggermente più a sinistra rispetto alla volta precedente.

LINEA 920: se le condizioni sono verificate, vuol dire che la freccia ha raggiunto il palloncino.

LINEA 950: come la linea 900

LINEA 970: come la linea 920

LINEA 1000: come la linea 900

LINEA 1020: come la linea 920

LINEA 1090÷1120: parte finale del programma (punteggio, richiesta di fine o proseguimento, ecc.).

LINEA 1130÷1270: cancellano l'omino col palloncino e al suo posto evidenziano l'omino a testa in giù che fanno poi cadere fino in fondo.

```
10 REM * ----------------------*
20 REM * DIFENSE PIG *          *
30 REM * ------ IBM PC ------- *
40      DIM BALL%(300):RANDOMIZE TIMER
50      SCREEN 1:CLS:COLOR 0,4:KEY OFF
60      FOR I=1 TO 9:CIRCLE(10,10),I,2:NEXT I
70      REM          'DRAW "p2,2"'
80      CIRCLE(5,25),5,3
90      DRAW"c3bm5,30g5be5f5c2u20c3bd20bh5d10g5be5f5"
100     CIRCLE(5,35),2,3
110     CIRCLE(14,35),1,1
120     GET (0,0)-(20,48),BALL%
130     CLS
140     DIM MAN%(100)
150     CIRCLE(5,20),5,2
160     DRAW"a2bm5,17g5be5f5bh5d10g5be5f5a0"
170     GET (0,0)-(20,25),MAN%
180     CLS
190     DIM BALC%(65)
200     GET (0,0)-(20,20),BALC%
210     CLS
220     DRAW"s02c3"
230     DRAW"bm2,10r56"
240     DRAW"b13d70br3156"
250     DRAW"br3u70br5f1416h2"
260     DRAW"15d12f5r14g7"
270     DRAW"br9132"
280     DRAW"d10r36b127bd26be24bu17br13f11"
290     LINE(5,46)-(12,33)
300     CIRCLE(16,10),14,3,-.01,-1.04,5/9
310     PAINT (20,9),1,3
320     CIRCLE(16,10),14,3,-1.04,-2.09,5/9
330     PAINT (16,6),2,3
340     CIRCLE(16,10),14,3,-2.09,-3.14,5/9
350     PAINT (10,9),1,3
360     CIRCLE(20,20),7.5,4.01,2.6
370     REM                 'DRAW"p3,3"'
380     CIRCLE(16,45),14,,-3.14,-4.18,5/9
390     PAINT (12,47),1,3
400     CIRCLE(16,45),14,,-4.19,-5.23,5/9
410     PAINT (16,49),2,3
420     CIRCLE(16,45),14,,-5.23,-6.28,5/9
430     PAINT (20,47),1,3
```

```
440   CIRCLE(10,28),21,2,1.88,4.5,12/5
450   DRAW"s02c1"
460   DRAW"bm12,26125e5bg5f5"
470   PAINT (20,20),2,3
480   PAINT (20,40),2,3
490   CIRCLE(17,17),1.3,0
500   DIM PIG%(300)
510   GET (0,0)-(34,58),PIG%
520   CLS
530   PER=0
540   IF PA=2 THEN VBD=171:MBD=-3
550   DRAW"s02c1"
560   DRAW"bm12,26125e5bg5f5"
570   DIM DARK%(25)
580   GET (0,0)-(18,30),DARK%
590   CLS
600   LINE (1,198)-(280,195),2,BF
610   LINE (0,0)-(319,199),1,B
620   D1=60:CF=0:L1=271:L2=271:L3=271:PA=0:PU=0:VBD=0:MBD=2
630   PA1=0
640   B1=INT(RND*250):BD=VBD
650   SP=INT((RND*55)+D1)
660   LINE (0,198)-(280,195),2,BF
670   BL1=B1+8:BL2=B1+16
680   PA1=PA1+1
690   IF PA=10 OR PA2<0 THEN PA2=10-PA:LOCATE 2,5:PRINT " PALLE
PASSATE " PA2:LOCATE 3,5:PRINT "      GAME OVER ":GOTO 1090
700   IF PA1=11 THEN LINE (1,150)-(280,194),0,BF:PA2=10-PA:COLOR
,1:FOR I=1 TO 2000:NEXT I:COLOR 8,1
710   IF PA1>10 THEN VBD=146:MBD=-3:BD=146:PA2=PA2-1
720   A$=INKEY$
730   BD=BD+MBD
740   BDF=BD+20
750   LOCATE 1,2:PRINT " PALLE " PA " FRECCE" PU "PUNTI "PER
760   IF BD>151 OR BD<0 THEN LINE (1,150)-(280,194),0,BF:GOTO 640
770   SPBD=BD+30
780   IF SPBD<SP THEN SPAS=B1+18
790   IF SPBD>SP THEN SPAS=SPAS+8:PSET (SPAS,SP),1:PSET (SPAS-8,SP),0
800   IF SPAS>285 AND SPAS<294 AND SP>D1+10 AND SP<D1+45 THEN GOTO
1280
810   PUT (B1,BD),BALL%,PSET
820   IF A$=" " AND CF<3 THEN GOTO 1040
830   IF A$="8" THEN D1=D1-4
840   IF A$="2" THEN D1=D1+4
850   IF D1<0 THEN D1=0
860   IF D1>140 THEN D1=140
870   PUT (284,D1),PIG%,PSET
880   IF CF<1 THEN GOTO 720
890   IF L1=1 THEN CF=CF-1:L1=L2:F1=F2:L2=L3:F2=F3:L3=271:LINE
(1,1)-(15,170),0,BF:GOTO 880
900   L1=L1-6:PUT (L1,F1),DARK%,PSET
910   FM=F1+5
```

```
920   IF L1>BL1 AND L1<BL2 AND FM>BD AND FM<BDF THEN LINE
(L1,F1)-(L1+18,F1+15),0,BF:L1=1:MOL=3:GOSUB 1130
930   IF CF<2 THEN GOTO 720
940   IF L2=1 THEN CF=CF-1:L2=L3:F2=F3:L3=271:GOTO 930
950   L2=L2-6:PUT (L2,F2),DARK%,PSET
960   FM=F2+5
970   IF L2>BL1 AND L2<BL2 AND FM>BD AND FM<BDF THEN LINE
(L2,F2)-(L2+18,F2+15),0,BF:L2=1:MOL=2:GOSUB 1130
980   IF CF<3 THEN GOTO 720
990   IF L3=1 THEN CF=CF-1:L3=271:GOTO 980
1000  L3=L3-6:PUT (L3,F3),DARK%,PSET
1010  FM=F3+5
1020  IF L3>BL1 AND L3<BL2 AND FM>BD AND FM<BDF THEN MOL=1:GOSUB 1130
1030  IF CF<4 THEN GOTO 720
1040  CF=CF+1
1050  PU=PU+1
1060  IF CF=1 THEN F1=D1+20:GOTO 880
1070  IF CF=2 THEN F2=D1+20:GOTO 880
1080  IF CF=3 THEN F3=D1+20:GOTO 880
1090  LOCATE 10,10:PRINT "Per continuare il gioco batti -Y-"
1100  LOCATE 15,11:INPUT B$
1110  IF B$="Y" OR B$="y" THEN COLOR 0,0:PER=0:GOTO 590
1120  END
1130  BEEP
1140  REM * PLAY"mbo3cde" *
1150  IF BD<40 THEN PER=PER+300*MOL:GOTO 1190
1160  IF BD<80 THEN PER=PER+200*MOL:GOTO 1190
1170  IF BD<120 THEN PER=PER+100*MOL:GOTO 1190
1180  PER=PER+50*MOL
1190  IF B1<140 THEN PER=PER+100*MOL
1200  PUT (B1,BD),BALC%,PSET
1210  BD=BD+20
1220  PA=PA+1
1230  FOR IND=BD TO 171
1240  PUT (B1,IND),MAN%,PSET
1250  NEXT IND
1260  BD=180
1270  RETURN
1280  PLAY"mbo2bao1fedc"
1290  FOR CAD=D1 TO 140 STEP 2
1300  PUT (284,CAD),PIG%,PSET
1310  NEXT CAD
1320  PA2=-1
1330  GOTO 690
```

BALL%

MAN%

PIG%

←

DAAR%

Olivetti M 20

Per questo elaboratore è valido tutto quanto già detto per l'I.B.M. P.C.

La dimensione della matrice che dovrà contenere il profilo generato viene però calcolata in un modo diverso, ed esattamente con la seguente formula:

$$\frac{B * H}{16} + 3 \qquad \text{(per video in B/N)}$$

$$\frac{B * H}{8} + 3 \qquad \text{(per video a colori)}$$

dove B è la base (in pixel) del rettangolo in cui è compreso il profilo, mentre H è l'altezza del rettangolo stesso.

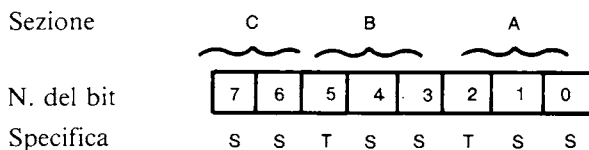
In più, in M20, un profilo può essere mosso all'interno di una finestra specificata, non solo all'interno dell'intero schermo video.

Anche il programma del maialino, che abbiamo elencato per l'I.B.M. P.C., con le dovute limitazioni e modifiche (per esempio CURSOR al posto di LOCATE), è valido anche per M 20.

APPLE II

È abbastanza complicato in questo elaboratore creare una tabella in memoria contenente la disposizione dei pixel che formano il profilo.

Ogni byte della memoria (riguardante il profilo) deve avere la seguente configurazione:

Sezione	C		B			A		
N. del bit	7	6	5	4	3	2	1	0
Specifica	S	S	T	S	S	T	S	S

dove le coppie di bit SS determinano lo spostamento e il bit T determina se il pixel deve essere acceso o spento. In particolare:

SS = 00 spostamento di un punto verso l'alto
SS = 01 spostamento di un punto verso destra
SS = 10 spostamento di un punto verso il basso
SS = 11 spostamento di un punto verso sinistra

T = 0 pixel spento (non tracciamento)
T = 1 pixel acceso (tracciamento)

Ogni sezione (A,B o C) riguarda quindi un solo punto luminoso della figura.
In più la sezione C può riguardare solo spostamenti di punti senza tracciamento, mancando il bit T. L'insieme delle informazioni che contengono la configurazione del profilo voluto devono essere registrate (utilizzando, ad esempio, il programma Monitor) in memoria, partendo da un indirizzo che si trovi in una zona che non venga cancellata da una istruzione HGR o HGR2 (per esempio l'indirizzo 1DFCH).
Ogni profilo viene numerato quindi da 1 a n, in modo che le istruzioni di grafica facciano riferimento a questo numero quando occorre. Le istruzioni grafiche riguardanti i profili sono ROT e SCALE, che devono precedere le istruzioni DRAW o XDRAW che disegnano il profilo, e il comando SHLOAD per caricare in memoria matrici di profili registrate su supporto esterno.
Supponiamo di aver creato, col procedimento precedente, un profilo e di averlo memorizzato nella prima posizione (quella riservata al profilo 1).
Il seguente programma evidenzia un modo di utilizzo del profilo creato.

Esempio:

```
10 HGR
20 HCOLOR = 7
30 FOR I = 1 TO 50
40 ROT = I
50 SCALE = I
60 DRAW 1 AT 130 + I, 70 + I
70 FOR K = 1 TO 1000 : NEXT K
80 XDRAW 1 AT 130 + I, 70 + I
90 NEXT I
```

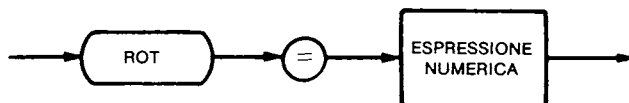
Risultato:

LINEA 10: definisce il modo per grafici ad alta risoluzione (280 x 160 pixel).
LINEA 20: determina che i grafici verranno disegnati in bianco.
LINEA 30: inizia il ciclo in cui sono inserite tutte le istruzioni grafiche.
LINEA 40: imposta una rotazione sempre crescente (da 1 a 50) ad ogni ripetizione.
LINEA 50: imposta la dimensione del profilo sempre maggiore (da 1 a 50) ad ogni ripetizione.
LINEA 60: disegna il profilo 1 inizialmente partendo dal punto di coordinate 131 e 71 e poi lo sposta ad ogni ripetizione in diagonale (+1 sull'asse X e +1 sull'asse Y).
LINEA 70: effettua un ritardo per dar modo allo spettatore di visualizzare il profilo.
LINEA 80: disegna il profilo nei colori complementari e nella stessa posizione del precedente: ciò equivale a cancellare il profilo dallo schermo.
LINEA 90: termine del ciclo iniziato alla linea 30.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

ROT (Istruzione)

Definisce la rotazione angolare per il profilo che verrà disegnato con un'istruzione DRAW o XDRAW.

Formalismo sintattico:

Esempio:

```
10 HGR
20 HCOLOR = 5
30 FOR I = 1 TO 40
40 ROT = I
50 DRAW 1 AT 100, 80
60 NEXT I
70 END
```

Risultato:

LA LINEA 10 cancella il video e lo predispone in modo grafico ad alta risoluzione.
LA LINEA 20 definisce il colore.
LA LINEA 30 innesca un ciclo.
LA LINEA 40 definisce l'angolo di rotazione.
LA LINEA 50 disegna il profilo 1 cominciando alla posizione (100, 80).
LA LINEA 60 fa terminare il ciclo iniziato alla linea 30.
LA LINEA 70 fa terminare l'esecuzione del programma.

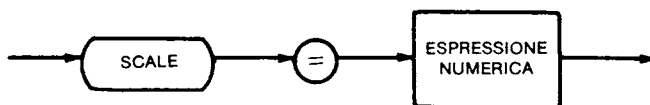
Nota:

— Il valore dell'espressione numerica può essere compreso tra 0 e 255. ROT = 0 farà disegnare il profilo esattamente come è stato definito, ROT = 32 farà in modo che il profilo venga disegnato capovolto, e così via.

SCALE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Definisce la dimensione di scala per il profilo che verrà disegnato con una istruzione DRAW o XDRAW.

Formalismo sintattico:

Esempio:

```
100 HGR
110 HCOLOR = 5 : A = 20
120 SCALE = A
130 DRAW 1 AT 100, 80
```

Risultato:

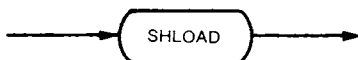
LA LINEA 100 cancella il video e lo predispone in modo grafico ad alta risoluzione.
LA LINEA 110 definisce il colore ed assegna il valore 20 alla variabile A.
LA LINEA 120 definisce la scala.
LA LINEA 130 disegna un profilo, definito precedentemente, alla posizione (110, 80) col fattore di scala impostato alla linea 120.

Note:
— Il valore dell'espressione·numerica può variare da 0 a 255.
— SCALE = 1 definisce la riproduzione punto per punto della definizione del profilo, SCALE = 255 ingrandisce di 255 volte ciascun vettore di tracciamento. SCALE = 0 è la dimensione massima e non un singolo punto.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

SHLOAD (Comando)

Carica una tabella di profili da nastro nella memoria di lavoro.

Formalismo sintattico:

Esempio:

```
50 SHLOAD
60 SCALE = 1 : ROT = 0 : COLOR = 5 : DRAW 1 AT 15, 15
```

Risultato:

LINEA 50: carica in memoria una tabella di profili e predispone il video in modo grafico ad alta risoluzione.

LINEA 60: definisce la scala, rotazione, colore e disegna la prima definizione nella tabella dei profili alla posizione (15,15).

CAPITOLO 11

GENERAZIONE SUONI

GENERALITA'

In molti elaboratori è presente un circuito speciale che, collegato ad un amplificatore audio e a un altoparlante, può produrre musica o rumori. Via software è normalmente possibile modificare la durata, la frequenza e il volume della musica emessa. Le modalità per l'emissione di suoni variano in dipendenza dell'elaboratore utilizzato.

TEXAS TI 99/4A

È possibile produrre tre suoni e un rumore contemporaneamente, ognuno con propria frequenza e volume, utilizzando l'istruzione CALL SOUND. L'estensione del suono è di 5 ottave, cioè da 110 Hz a 44.733 Hz; il suono viene inviato al televisore assieme al segnale video.

La durata, valida per tutti i tre toni contemporanei, è data in millisecondi, e può variare da —4250 a 4250 ms.

Normalmente una CALL SOUND attende la fine dell'esecuzione di una CALL SOUND precedentemente eseguita, a meno che non sia data una durata negativa.

La frequenza viene data direttamente in Hertz. Se la frequenza è negativa (da —1 a —8) vengono emessi 4 tipi diversi di rumore rosa (periodico) e 4 tipi diversi di rumore bianco.

Il volume deve essere un numero compreso tra 0 (il più alto) e 30 (il più basso).

COMMODORE VIC 20

Il VIC 20 ha cinque locazioni di memoria, modificabili tramite POKE, che controllano il suono in uscita. Modificando opportunamente tali locazioni è possibile produrre tre suoni e un rumore contemporaneamente, con lo stesso volume per tutti e quattro. Il suono viene inviato al televisore assieme al segnale video.

Le locazioni di memoria interessate sono:
36874_D registro toni bassi (da 31 Hz a 3995 Hz)
36875_D registro toni medi (da 63 a 7990 Hz)
36876_D registro toni alti (da 125 Hz a 15980 Hz)
36877_D registro del rumore
36878_D registro del volume (0 il più basso, 15 il volume massimo).

I valori utilizzabili nella POKE a queste locazioni di memoria vanno, per le prime quattro locazioni, da 128 (per le frequenze più basse) fino a 254 (per le frequenze più alte). Per esempio, il DO centrale (circa 262 Hz) è dato da 240 per i toni bassi, da 225 per i toni medi e da 194 per i toni alti. Mediante accorgimenti vari è possibile creare i suoni più disparati, dal vibrato al tremulo, a suoni speciali da utilizzare in concomitanza di videogiochi, ritmi vari ecc.

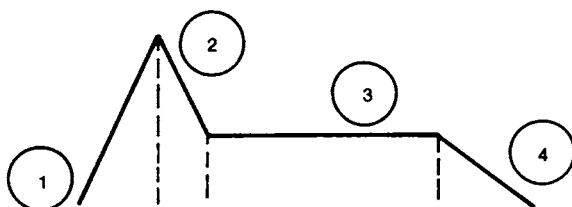
Una volta impostato il registro opportuno, l'altoparlante continua a suonare all'infinito.

Quando si vuole far cessare la musica è necessario azzerare il registro del volume.

CBM 64

Il CBM 64 è dotato di un sofisticato sintetizzatore elettronico musicale, dotato di tre voci completamente indipendenti che coprono 8 ottave, un generatore di inviluppo, filtratura, modulazione e generatore di rumore bianco. Il suono viene inviato al televisore assieme al segnale video.

Il circuito sintetizzatore (un 6581) è composto da ben 29 registri, di cui 25 modificabili attraverso istruzioni POKE.

Per ogni voce (oscillatore) vi sono due registri (2 * 8 = 16 bit) per il valore della frequenza, due registri (per un totale di 12 bit utili) per la determinazione della forma d'onda, un registro di controllo per la determinazione del tipo di forma d'onda (triangolare, rettangolare, a dente di sega, rumore), un registro per l'attaccare/decomporre, un registro per il sostenere/rilasciare. Un suono è infatti suddivisibile in quattro fasi:

Nella fase 1 il volume passa da zero al suo valore di picco (fase di ATTACK = attaccare), successivamente la nota scende di volume dal valore di picco ad un valore medio (fase 2 o di DECAY = decomporre). Il livello medio raggiunto, che viene poi mantenuto costante per la maggior parte del tempo della nota, è detto SUSTAIN (= sostenere, fase 3). Quando, infine, la nota cessa di suonare, passa dal valore medio al valore zero (fase 4 o di RELEASE = rilasciare).

Ognuna di queste fasi è programmabile, utilizzando quattro bit (presi dagli ultimi due registri).

Il tempo della fase ATTACK varia da 2 ms a 8 sec. per un valore decimale da 0 a 16 da inserire nei 4 bit. Il tempo delle fasi DECAY/RELEASE variano da 6 ms a 24 sec. per un valore da 0 a 16 da inserire nei 4 bit.

Vi sono poi altri 4 registri: due (con 11 bit utili) per impostare la frequenza di taglio, uno per attivare i filtri per i tre oscillatori, l'ultimo, diviso in una coppia di 4 bit, per determinare il tipo di filtro (passa-basso, passa-alto, passa-banda) e per la determinazione del volume.

Segue l'elenco di tutti i registri interessati dal sintetizzatore, con indicazione della posizione relativa al primo

VOCE 1

54272_D	Frequenza	(Byte basso)	0
54273_D	Frequenza	(Byte alto)	+1
54274_D	Forma d'onda	(Byte basso)	+2
54275_D	Forma d'onda	(Byte alto)	+3
54276_D	Tipo di forma d'onda		+4
54277_D	ATTACK/DECAY		+5
54278_D	SUSTAIN/RELEASE		+6

VOCE 2

54279_D	Frequenza	(Byte basso)	+7
54280_D	Frequenza	(Byte alto)	+8
54281_D	Forma d'onda	(Byte basso)	+9
54282_D	Forma d'onda	(Byte alto)	+10
54283_D	Tipo di forma d'onda		+11
54284_D	ATTACK/DECAY		+12
54285_D	SUSTAIN/RELEASE		+13

VOCE 3

54286_D	Frequenza	(Byte basso)	+14
54287_D	Frequenza	(Byte alto)	+15
54288_D	Forma d'onda	(Byte basso)	+16
54289_D	Forma d'onda	(Byte alto)	+17
54290_D	Tipo di forma d'onda		+18
54291_D	ATTACK/DECAY		+19
54292_D	SUSTAIN/RELEASE		+20

FILTRI

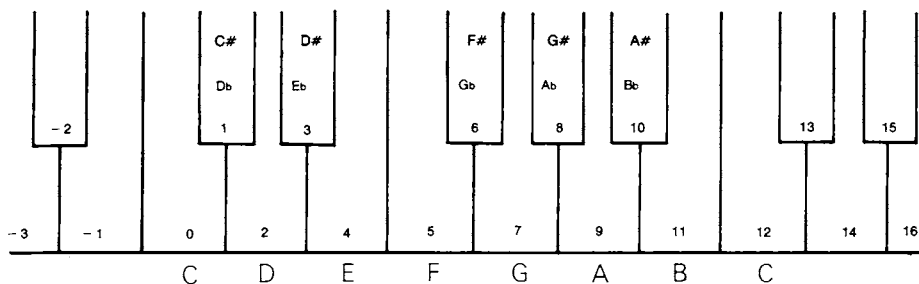
54293_D	Frequenza di taglio	(Byte basso)	+21
54294_D	Frequenza di taglio	(Byte alto)	+22
54295_D	Attivazione filtri		+23
54296_D	Tipo filtro + volume		+24

Con un uso oculato e accorto di tali registri è possibile generare non solo dell'ottima musica, ma anche degli interessanti effetti sonori per qualsiasi tipo di applicazione.

SINCLAIR ZX SPECTRUM

Esiste un solo altoparlante, interno all'elaboratore, con cui è possibile eseguire della musica utilizzando l'istruzione BEEP. La qualità e il volume dell'altoparlante incorporati sono abbastanza scarsi, per cui è possibile utilizzare il segnale audio presente sulle prese "MIC" e "EAR" e inviarlo a un registratore, oppure ad un impianto di alta fedeltà, o ad una cuffia, ecc.

Nell'istruzione BEEP la durata è espressa in secondi, mentre la frequenza è data come altezza in semitoni sopra il DO centrale, con numeri positivi per i semitoni superiori e negativi per i semitoni inferiori.

Esempio (tastiera di una ottava di pianoforte):

Ricordarsi che:
C = DO, D = RE, E = MI, F = FA, G = SOL,
A = LA, B = SI, # = diesis, b = bemolle

Se si volessero intervalli più brevi o più lunghi di un semitono, è possibile modificare, attraverso una POKE, l'indirizzo di memoria 23609D, che contiene appunto l'informazione su tale intervallo.

SHARP MZ 700

È dotato di un altoparlante interno con volume regolabile solo manualmente. Il segnale audio non viene inviato al televisore. La gamma di frequenze copre tre ottave (quella centrale, la più bassa e la più alta). Le istruzioni BASIC interessate sono MUSIC per specificare la melodia e TEMPO per determinare la velocità di esecuzione.

Nell'istruzione MUSIC è necessario specificare tutte le note che devono essere eseguite. Per ogni nota si possono specificare tre qualità:

1) Specifica dell'ottava: niente per l'ottava centrale, — per quella più bassa, + per quella più alta.

2) Specifica della nota (eventualmente preceduta dal simbolo # per i diesis):
   C per il DO
   D per il RE
   E per il MI
   F per il FA
   G per il SOL
   A per il LA
   B per il SI
   R per la pausa (intervallo di silenzio)
3) Specifica della durata della nota o della pausa: un numero da 0 a 9 col seguente significato:
   0 = nota o pausa di 1/32
   1 = nota o pausa di 1/16
   2 = nota o pausa di 1/16 puntata
   3 = nota o pausa di 1/8
   4 = nota o pausa di 1/8 puntata
   5 = nota o pausa di 1/4
   6 = nota o pausa di 1/4 puntata
   7 = nota o pausa di 1/2
   8 = nota o pausa di 1/2 Puntata
   9 = nota o pausa intera

Se non viene specificata la durata, si assume 1/4. Per la velocità di esecuzione viene usata l'istruzione TEMPO seguita da una espressione numerica con valore da 1 a 7: 1 per la velocità più lenta (adagio), 7 per velocità più alta (molto allegro). Se non viene specificata l'istruzione TEMPO viene assunto il valore 4 (tempo medio, moderato).

IBM P.C.

Ha un solo altoparlante, con volume non modificabile via software. Esistono tre istruzioni orientate alla musica: BEEP, SOUND e PLAY.

Il BEEP fa suonare l'altoparlante a 800 Hz per 1/4 di secondo, e serve quindi esclusivamente come campanello di avviso (per esempio, in concomitanza di errori rilevati in fase di esecuzione di un programma).

L'istruzione SOUND permette di generare musica definendo la frequenza (direttamente in Hertz) e la durata (in battiti dell'orologio, cioè circa ogni 1/18 di secondo) della nota emessa. La frequenza può essere compresa fra 37 e 32767 Hz, mentre la durata può andare da 0 a 65535.

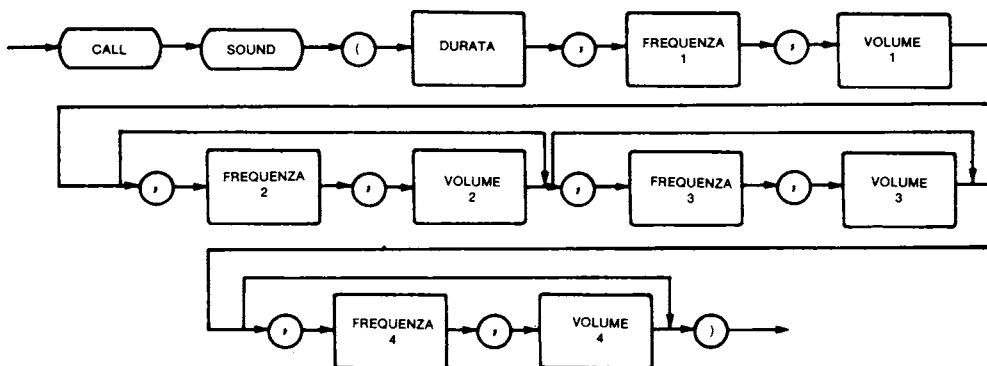
Mentre un'istruzione SOUND è in corso, il programma continua l'esecuzione. Se viene incontrata un'altra istruzione SOUND, il programma attende il completamento della precedente. È necessario dare una istruzione SOUND per ogni nota voluta.

L'istruzione PLAY è notevolmente più complessa, e permette di generare, attraverso una stringa di caratteri, un certo numero di note diverse. Nella stringa è possibile inserire un elevato numero di comandi con funzioni diverse, dai caratteri A ÷ G per la definizione delle note (dal LA al SOL), a + per il diesis, — per il bemolle, O per definire una delle 7 ottave (0 ÷ 6), L per definire la lunghezza della nota, P per definire una pausa, T per definire il tempo, X seguito da una variabile per poter inserire il contenuto di altre variabili stringa, e così via. Ognuno di questi comandi, e l'eventuale numero che segue il comando, possono essere inseriti nella stringa di PLAY senza separatori oppure con un punto e virgola tra loro. Eventuali spazi vengono ignorati. È possibile inserire delle variabili usando VARPTR$ al posto del comando X. Vedere anche Appendice A.

CALL SOUND (istruzione)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Il sottoprogramma SOUND dice al TI 99/4A di produrre suoni a diversa frequenza, specificando tre caratteristiche di ogni suono.

Formalismo sintattico:

Esempio:

```
100 FOR T = 1 TO 10
110 CALL SOUND (- 500, T * 110, 1)
120 NEXT T
130 FOR T = 1 TO 10
140 CALL SOUND (+ 500, T * 110, 1)
150 NEXT T
160 CALL SOUND (500, - 1, 2)
170 END
```

Risultato:

LINEE 100÷ 120:emettono 10 note molto velocemente, poichè la durata è negativa. Le note vanno dal LA minore (110 Hz) al DO diesis due ottave sopra il DO centrale (circa 1100 Hz).

LINEE 130÷ 150:emettono 10 note con un intervallo di mezzo secondo tra loro. Le note sono le stesse viste in precedenza.

LINEA 160: emette un rumore periodico (rumore rosa) per mezzo secondo.

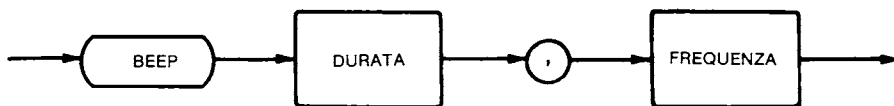
Note:

— DURATA è una espressione numerica che può assumere valori compresi tra 1 e 4250 oppure tra —1 e —4250.
Indica la durata effettiva della nota in millisecondi. Se è negativa, il nuovo suono non attende la fine di quello in corso, ma lo interrompe.
— FREQUENZA 1, 2, 3, 4 sono espressioni numeriche che possono assumere valori compresi tra 110 e 44.733. La frequenza è data direttamente in Hertz (periodi per secondo dell'onda sonora generata). Possono anche assumere valori negativi compresi tra —1 e —8, e allora verranno generati rumori periodici (o rumori rosa: valori tra —1 e —4) oppure rumori bianchi (valori tra —5 e —8).
— VOLUME 1, 2, 3, 4 sono espressioni numeriche che possono assumere valori compresi tra 0 (volume più alto) e 30 (volume più basso).
— Possono essere generati contemporaneamente tre suoni diversi e un rumore.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

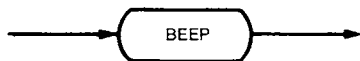
BEEP (Istruzione)

In IBM P.C. fa suonare l'altoparlante a 800 Hz per 1/4 di secondo. Nello ZX Spectrum può produrre suoni con frequenza e durata voluti.

Formalismo sintattico: per ZX Spectrum

Esempio:

```
10 LET A = 1
20 BEEP A/8, 20 : BEEP A/16, 18 : BEEP A/4, 20
30 BEEP A/8, 14 : BEEP A/4, 8 : BEEP A/4, 14 : BEEP A/16, 18
40 BEEP A/4, 8 : BEEP A/8, 14 : BEEP A/4, 8
50 BEEP A/4, 14 : BEEP A/16, 18 : BEEP A/4, 8 : BEEP A/8, 14
60 BEEP A/16, 18 : BEEP A/4, 8 : BEEP A/4, 14
70 BEEP A/4, 15 : BEEP A/4, 18 : BEEP A/4, 19 : BEEP A/8, 20
```

Risultato:
Alla riga 10 viene assegnato il valore 1 alla variabile A, per dare alla nota intera il valore di un secondo. Quindi un BEEP di durata A/4, sarà una nota di 1/4 di secondo.
Le righe dalla 20 alla 70 definiscono la melodia del tema iniziale della serenata in RE maggiore di Beethoven, opera 25.

Formalismo sintattico per: IBM P.C.

Note:
— DURATA e FREQUENZA sono espressioni numeriche.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

MUSIC (Istruzione)

Emette all'altoparlante la melodia o gli effetti sonori specificati dall'espressione stringa. La velocità con cui viene eseguita questa melodia è quella specificata con l'istruzione TEMPO.

Formalismo sintattico:

Esempio:

```
10 A$ = "- G1R - GR - GR - # D9R1"
20 B$ = "- FR - FR - FR - D9"
30 TEMPO 6
40 MUSIC A$, B$
50 END
```

Risultato:

LINEE 10÷20: assegnano la stringa specificata alle variabili A$ e B$.
LINEA 30: determina il tempo con cui deve essere suonata la melodia (allegro).
LINEA 40: suona la melodia presente nelle due variabili stringa A$ e B$. In questo caso si tratta della parte iniziale della nona sinfonia di Beethoven.
LINEA 50: fa terminare l'esecuzione del programma.

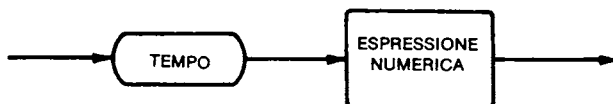
Note:

— Per le modalità di uso dell'espressione stringa, vedere il paragrafo "generalità" di questo stesso capitolo.
— Il volume del suono prodotto non è modificabile da programma.

TEMPO (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Questa istruzione determina il tempo al quale viene suonata la musica mediante l'istruzione MUSIC.
Se non esiste tale istruzione, si assume il TEMPO 4.

Formalismo sintattico:

Esempio:

```
10 A$ = "+ A3 + # F1 + A + B3A + D+ # F1A"
20 B$ = "+ D3A + D + # F1A + D3 + # F1A"
30 C$ = "+ D + E + # F + G + A3R"
40 TEMPO 1
50 MUSIC A$, B$, C$
60 END
```

Risultato:

LINEE 10÷30: assegnano la stringa specificata alle variabili A$, B$ e C$.
LINEA 40: determina il tempo con cui deve essere suonata la melodia (lento, adagio).
LINEA 50: suona la melodia presente nelle tre variabili stringa A$, B$ e C$. In questo caso si tratta della parte iniziale della serenata in re maggiore, opera 25, di Beethoven.
LINEA 60: fa terminare l'esecuzione del programma.

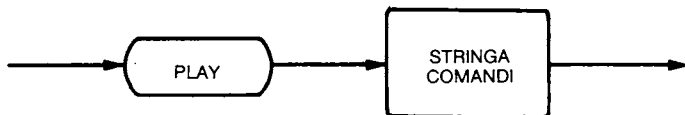
Note:

— Per maggiori informazioni vedere il paragrafo "generalità" in questo stesso capitolo.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

PLAY (Istruzione)

Suona la musica specificata nell'espressione stringa.

Formalismo sintattico:

Esempio:

```
10 PLAY "MNT140O4L8AL16F#L4AL8DO3L4AO4L4DL16F#"
20 PLAY "O3L4AO4L8DO3L4AO4L4DL16F#"
30 PLAY "O3L4AO4L8DL16F#O3L4AO4L4DEF#GL8A"
```

Risultato:

Viene svolto il tema iniziale della serenata in RE maggiore di Beethoven, opera 25.

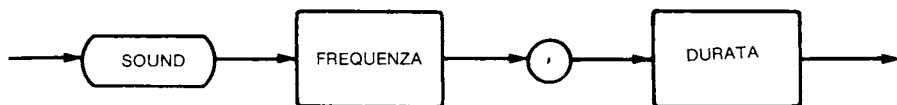
Note:

— STRINGA COMANDI identifica tutta una serie di comandi che costituiscono un vero e proprio linguaggio interno. Vedere l'appendice A per maggiori ragguagli (alla voce "STRINGA COMANDI").

SOUND (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Genera il suono attraverso l'altoparlante.

Formalismo sintattico:

Esempio:

```
10 FOR A = 220 TO 1760 STEP 10
20 SOUND A, 1
30 NEXT
40 FOR A = 1760 TO 220 STEP - 10
50 SOUND A, 1
60 NEXT
```

Risultato:
Crea una serie di note che, partendo dal "LA" a 220 Hz sale di 3 ottave e poi ridiscende di nuovo fino al "LA" a 220 Hz.

Nota:
— FREQUENZA e DURATA sono espressioni numeriche. La frequenza è data in Hertz (cicli al secondo).

CAPITOLO 12

GESTIONE DEGLI ERRORI

VARIABILI DI SISTEMA

ERL

ERR (ERN)

ISTRUZIONI

ERROR

IF ERL ...

IF ERR (o ERN) ...

ON ERROR GOTO

RESUME

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

ERL e ERR (Variabili)

Ritornano rispettivamente il numero della riga e il codice dell'ultimo errore riscontrato.

Possono essere impostate usando l'istruzione ERROR.

Normalmente vengono utilizzate in istruzioni IF per dirigere il flusso di programma nella zona di programma incaricata delle segnalazioni (con eventuale recupero) degli errori riscontrati.

In MZ 700 al posto della ERR si usa ERN.

Esempio (su MZ 700):

```
 100 ON ERROR GOTO 1000
 110 NEXT I
 120 RETURN
 130 PRINT "PROSEGUIMENTO SENZA ERRORI"
     :
 990 END
1000 IF (ERN = 13) * (ERL = 110) RESUME 120
1010 PRINT "ERRORE"; ERN; "ALLA LINEA"; ERL
1020 RESUME NEXT
```

Risultato:

LA LINEA 100 prepara il sistema a un salto alla linea 1000 in caso di intercettamento di un errore.

Alla linea 110 è presente un'istruzione NEXT senza FOR, che origina un errore. ERN conterrà 13 e ERL conterrà 110. Il controllo viene spostato alla linea 1000, nella quale l'insieme delle condizioni legate da AND risulta verificato ed il controllo spostato alla linea 120. Su tale linea è presente una istruzione RETURN che non fa riferimento a nessuna istruzione GOSUB, per cui viene originato un errore 14. Il controllo passa alla linea 1000, il cui insieme di condizioni risulta ora falso, per cui viene eseguita la linea successiva, la 1010, che produce la seguente stampa:

ERRORE 14 ALLA LINEA 120

Dopodichè viene eseguita la linea 1020, che riporta il controllo alla linea immediatamente successiva a quella che ha dato origine all'errore, cioè, in questo caso, sposta il controllo alla linea 130.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

ERROR (Istruzione)

Simula il ripetersi di un errore BASIC, oppure consente di definire i propri codici di errore.

Se il valore dell'espressione numerica è uguale a un codice di errore usato dal BASIC, simula l'errore stesso, altrimenti definisce un nuovo codice di errore, che può essere poi utilizzato allo stesso modo degli altri.

Formalismo sintattico:

Esempio:

```
  10 ON ERROR GOTO 1000
  20 INPUT "DAMMI LA TUA ETA'"; E
  30 IF (E < 5) OR (E > 100) THEN ERROR 200
     :
1000 IF ERR = 200 THEN PRINT "ETA' ERRATA"
1010 IF ERL = 30 THEN RESUME 20
1020 END
```

Risultato:

LINEA 10: prepara il sistema ad un salto alla linea 1000 in caso di intercettamento di un errore.

LINEA 20: attende un "input" numerico da tastiera, che assegna poi alla variabile numerica E.

LINEA 30: controlla che E sia compresa tra 5 e 100. Se così non fosse, definisce un nuovo codice di errore (200). Il controllo, a causa dell'istruzione alla linea 10, viene trasferito alla linea 1000.

LINEA 1000: stampa il messaggio specificato se nella variabile di sistema ERR è presente un codice di errore 200.

LINEA 1010: riporta il controllo dell'esecuzione alla linea 20 se nella variabile di sistema ERL è presente un valore 30, altrimenti viene eseguita la linea 1020 che fa terminare l'esecuzione del programma.

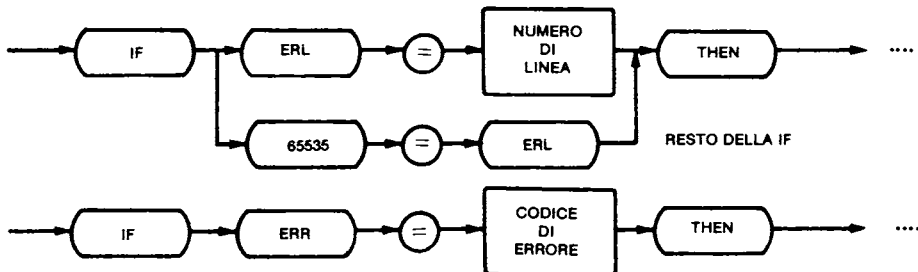
Nota:

— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 255.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

IF ERL ...
IF ERR ... (Istruzioni)
(IF ERN) ...

Consentono di testare le due variabili di errore. Se l'istruzione che ha causato l'errore è stata data in modo immediato, la variabile ERL conterrà 65535.
In MZ 700 al posto di ERR si usa ERN.

Formalismo sintattico:

Esempio (su MZ 700):

```
 100 ON ERROR GOTO 1000
 110 NEXT I
 120 RETURN
 130 PRINT "PROSEGUIMENTO SENZA ERRORI"
     :
 990 END
1000 IF (ERN = 13) * (ERL = 110) RESUME 120
1010 PRINT "ERRORE"; ERN; "ALLA LINEA"; ERL
1020 RESUME NEXT
```

Risultato:

LA LINEA 100 prepara il sistema ad un salto alla linea 1000 in caso di intercettamento di un errore.
Alla linea 110 è presente una istruzione NEXT senza FOR, che origina un errore. ERN conterrà 13 e ERL conterrà 110. Il controllo dell'esecuzione viene spostato alla linea 1000, nella quale l'insieme delle condizioni legate da AND (*) risulta verificato ed il controllo dell'esecuzione viene spostato alla linea 120. Su tale linea è presente una istruzione RETURN che non fa riferimento a nessuna istruzione GOSUB, per cui viene originato un errore 14. Il controllo dell'esecuzione passa alla linea 1000, il cui insieme di condizioni risulta ora falso, per cui viene eseguita la linea successiva, la 1010, che produce la seguente stampa:

ERRORE 14 ALLA LINEA 120

Dopodichè viene eseguita la linea 1020, che riporta il controllo dell'esecuzione alla linea immediatamente successiva a quella che ha dato origine all'errore, e cioè, nel nostro caso, sposta il controllo dell'esecuzione alla linea 130.

Note:
- NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65535.
- CODICE DI ERRORE è un'espressione numerica che può assumere valori compresi fra 0 e 255.

ON ERROR (Istruzione)

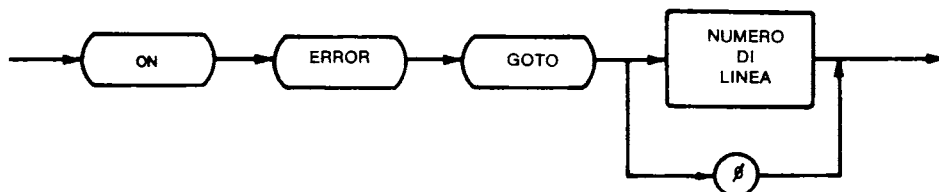
	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

Consente l'intercettazione degli errori e specifica la prima riga del sottoprogramma di gestione degli errori.

Se viene specificato 0 come numero di riga, viene disabilitata l'intercettazione degli errori, per cui, in caso di errore, il BASIC interromperà l'esecuzione del programma.

Formalismo sintattico:

Esempio:

```
  10 ON ERROR GOTO 1000
  20 INPUT "DAMMI LA TUA ETA'"; E
  30 IF (E < 5) OR (E > 100) THEN ERROR 200
   :
1000 IF ERR = 200 THEN PRINT "ETA' ERRATA"
1010 IF ERL = 30 THEN RESUME 20
1020 END
```

Risultato:

LINEA 10: prepara il sistema ad un salto alla linea 1000 in caso di intercettamento di un errore.

LINEA 20: attende un "input" da tastiera che assegna poi alla variabile E.

LINEA 30: controlla che E sia compresa fra 5 e 100. Se così non fosse, definisce un nuovo codice di errore (200). Il controllo dell'esecuzione, a causa dell'istruzione alla linea 10, viene trasferito alla linea 1000.

LINEA 1000: stampa il messaggio specificato se nella variabile di sistema ERR è presente un codice di errore 200.

LINEA 1010: riporta il controllo dell'esecuzione alla linea 20 se nella variabile di sistema ERL è presente un valore 30, altrimenti viene eseguita la linea 1020 che fa terminare l'esecuzione del programma.

Variante per: Apple

Nota:

— NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65.535.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

RESUME (Istruzione)

Continua l'esecuzione del programma dopo una procedura di ripristino dell'errore. Se non viene specificato niente, o 0, l'esecuzione riprende dall'istruzione che ha causato l'errore. Se viene specificato NEXT l'esecuzione riprende dall'istruzione successiva a quella che ha causato l'errore. Se si specifica un numero di linea, l'esecuzione riprende dal numero di linea specificato.

Formalismo sintattico:

Esempio:

```
   10 ON ERROR GOTO 1000
   20 INPUT "DAMMI LA TUA ETA'"; E
   30 IF (E < 5) OR (E > 100) THEN ERROR 200
      :
 1000 IF ERR = 200 THEN PRINT "ETA' ERRATA"
 1010 IF ERL = 30 THEN RESUME 20
 1020 END
```

Risultato:

LINEA 10: prepara il sistema ad un salto alla linea 1000 in caso di intercettamento di un errore.

LINEA 20: attende un "input" numerico da tastiera, che assegna poi alla variabile numerica E.

LINEA 30: controlla che E sia compresa tra 5 e 100. Se così non fosse, definisce un nuovo codice di errore (200). Il controllo dell'esecuzione, a causa dell'istruzione alla linea 10, viene trasferito alla linea 1000.

LINEA 1000: stampa il messaggio specificato se nella variabile di sistema ERR è presente un codice di errore 200.

LINEA 1010: riporta il controllo dell'esecuzione alla linea 20 se nella variabile di sistema ERL è presente un valore 30, altrimenti viene eseguita la linea 1020 che fa terminare l'esecuzione del programma.

Variante per: Apple

Nota:

— NUMERO DI RIGA è un'espressione numerica che può assumere valori compresi fra 0 e 65535.

CAPITOLO 13

GESTIONE DEI FILE

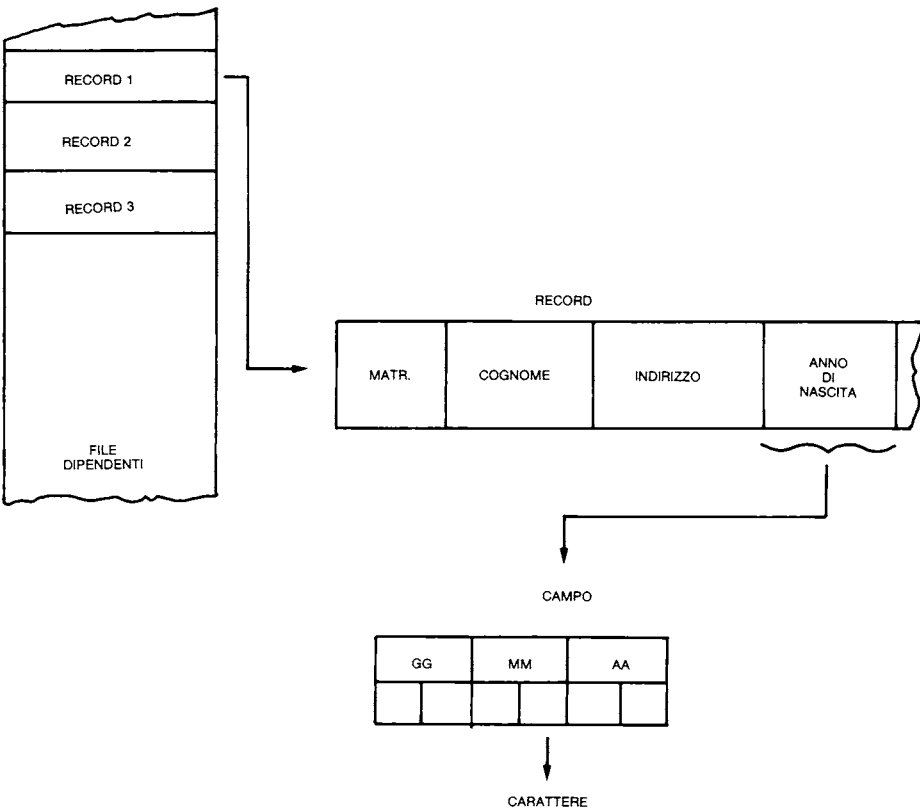
GENERALITA'

I mezzi più comuni per memorizzare informazioni nei piccoli elaboratori sono i nastri magnetici e i floppy disk.

Nei nastri magnetici le informazioni possono essere memorizzate solo in modo sequenziale (cioè un record dopo l'altro), mentre coi dischi si può accedere alle informazioni sia in modo sequenziale che in modo diretto.

Le informazioni vengono memorizzate sui supporti esterni, organizzate in FILE. Un "file" è quindi un insieme di informazioni omogenee che riguardano un particolare argomento. (Per esempio: una agenda telefonica, i titoli dei libri di una biblioteca, gli articoli di un magazzino, i dati dei dipendenti di una azienda, ecc.).

Vi sono normalmente due tipi di file: file di programmi e file di dati. I file di programmi contengono le istruzioni del linguaggio di programmazione utilizzato (BASIC o linguaggio macchina).

I file di dati possono essere a loro volta visti come un insieme di oggetti tra loro omogenei (RECORD). Per esempio, se il file in oggetto riguarda i dipendenti di un'azienda, tutte le informazioni relative ad un dipendente costituiscono un RECORD.

Ogni record è a sua volta costituito da CAMPI.

Nell'esempio precedente, se le informazioni riguardanti un dipendente costituiscono un record, si può considerare che la matricola, il cognome, l'indirizzo, ecc. possono essere considerati tanti campi diversi dello stesso record.

Ogni campo può a sua volta essere suddiviso in sòttocampi, o campi a livello inferiore. Nell'esempio, il campo ANNO DI NASCITA è costituito dai sottocampi GIORNO, MESE e ANNO, ognuno formato da due caratteri.

Ogni file presente sul supporto esterno è identificato mediante un nome opportuno.

FILE SEQUENZIALI

Quando in un file i record vengono registrati uno dopo l'altro, in sequenza, tali file vengono chiamati "SEQUENZIALI". Tali file possono essere riletti solo in modo sequenziale. Non è possibile modificarli direttamente, ma è necessario leggere un record per volta, eventualmente modificarlo in memoria, e riscriverlo poi in uscita; per tale operazione servono però due unità di uscita collegate contemporaneamente all'elaboratore. Questi file vengono normalmente creati in modo che ogni record che li compone sia ordinato (in modo crescente o decrescente) su un opportuno campo, che in tal caso viene chiamato "chiave" del file.

Nell'esempio precedente, il campo chiave potrebbe essere costituito dalla matricola del dipendente.

FILE RANDOM

Nei file di questo tipo i record vengono registrati in ordine qualsiasi. Possono essere letti e modificati in modo diretto, richiedendo direttamente il record interessato e senza quindi dover leggere (e riscrivere) tutti quelli che lo precedono. Anche per questo vengono detti ad "accesso diretto". Possono però anche venir letti in modo sequenziale.

Per la lettura del record interessato è necessario conoscere la "chiave" di accesso al record stesso, che deve essere univoca in tutto l'archivio.

ISTRUZIONI RELATIVE ALLA GESTIONE DEI FILE

In un programma BASIC, prima di poter accedere, con opportune istruzioni, alle informazioni contenute in un file, esso deve prima essere "aperto" mediante un'istruzione OPEN.

Al termine dell'utilizzo, il file deve essere "chiuso" mediante un'istruzione CLOSE.

Poichè nello stesso programma possono venir trattati più file contemporaneamente, è necessario che, in tutte le istruzioni che riguardano i file, sia presente un'informazione che specifichi il file cui si riferisce. Tale informazione è il "numero del file".

Altre istruzioni stabiliscono il tipo del file (sequenziale o random), se il file è in ingresso o in uscita, cioè se è da leggere o da scrivere. Altre istruzioni sono utilizzate per leggere e portare quindi in memoria i record dei file oppure per registrare le informazioni, presenti in memoria, sul supporto esterno.

Non tutti gli elaboratori trattano i file nello stesso modo.

Normalmente, nella versione del BASIC presente nell'elaboratore all'atto dell'acquisto, non esistono tutte le istruzioni atte alla gestione dei file, ma esse vengono aggiunte all'atto dell'acquisto delle unità periferiche (specialmente per quanto riguarda le unità disco). Vediamo, elaboratore per elaboratore, come viene effettuata la gestione dei file con la versione base dell'interprete BASIC.

TEXAS TI 99/4A

Gestisce sia file sequenziali che file random.

I record possono essere a lunghezza fissa oppure a lunghezza variabile. È possibile gestire fino a 3 (o fino a 9 con l'istruzione CALL FILES) file nello stesso programma, di cui 2 su nastro.

Potendo collegare due registratori contemporaneamente, è possibile effettuare modifiche anche a file sequenziali.

Non ha istruzioni differenziate per la lettura (o scrittura) di file sequenziali o ad acesso diretto.

COMMODORE VIC 20 e 64

Gestiscono sia file sequenziali che file random.

I record possono essere sia a lunghezza fissa che variabile.

Non hanno istruzioni differenziate per la lettura (o scrittura) di file sequenziali o ad accesso diretto.

SINCLAIR ZX 81

Non gestisce nessun tipo di file su supporto esterno.

SINCLAIR ZX SPECTRUM

È possibile gestire file sequenziali e random, utilizzando come supporti esterni il registratore (poco indicato) o delle opportune unità periferiche speciali (microdrive).

SHARP MZ 700

Nella versione base accetta solo file sequenziali su nastro. In un programma può essere definito un solo file.

OLIVETTI M20

Non ha unità periferiche a nastri, ma solo dischi magnetici. Può gestire file sequenziali e file ad accesso diretto. È possibile gestire fino a 15 file contemporaneamente in un programma BASIC. I record possono essere a lunghezza fissa (sia sequenziale che random) o variabile (solo sequenziale). Esistono istruzioni di lettura e scrittura che si differenziano per file sequenziali o per file random.

IBM P.C.

Oltre ai minidischi ha anche una unità registratore a cassetta. Può gestire file sequenziali e random. I record possono essere a lunghezza fissa o variabile.

Esistono istruzioni di lettura e scrittura che si differenziano per i file sequenziali da quelle per i file random.

APPLE II

Oltre a cassette magnetiche è possibile collegare unità a minidischi.

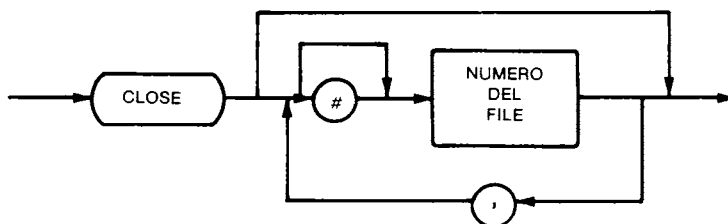
Non esistono nel BASIC istruzioni per la gestione dei file. È possibile però, attraverso artifici, richiamare da BASIC tutti i comandi del D.O.S. (Disk Operating System) che trattano appunto i file su disco.

Vedere l'Appendice C per l'elenco di tutti i comandi D.O.S. che interessano la gestione dei file.

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

CLOSE (Istruzione)

Chiude i file dati.

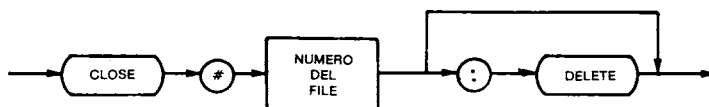
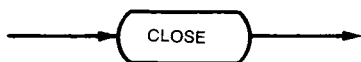
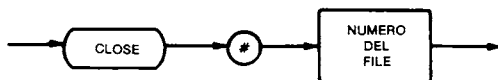
Formalismo sintattico:

Esempio:

```
10 OPEN "PIPPO" AS 1
20 READ # 1; NOME$, VIA$, CITTA'$, TELEF
30 CLOSE 1
```

Risultato:

LA LINEA 10 apre il file PIPPO con il numero di file 1.
LA LINEA 20 effettua una lettura sul file 1.
LA LINEA 30 chiude il file PIPPO.

Variante per: VIC 20 - CBM 64

Variante per: TI 99/4A

Variante per: MZ 700

Variante per: ZX Spectrum

Note:

— L'esecuzione di alcune istruzioni (END, NEW, RUN, ecc.) provoca la chiusura automatica di tutti i file ancora aperti nel programma.

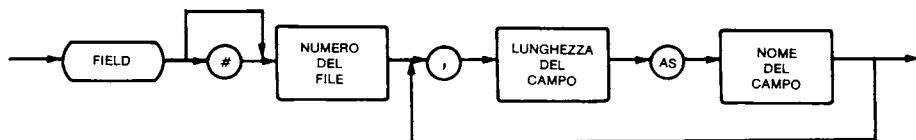
— In M20 e P.C. IBM una istruzione CLOSE senza numero del file provoca la chiusura di tutti i file aperti.

— NUMERO DEL FILE è un'espressione numerica i cui valori devono essere compresi fra 1 e 3 per IBM P.C., fra 1 e 15 per M20, fra 4 e 15 per ZX Spectrum, fra 1 e 127 per VIC 20 e CBM 64 e fra 0 e 255 per TI 99/4A.

FIELD # (Istruzione)

	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Definisce i campi nel buffer di I/O per un file ad accesso diretto. Il nome del campo deve essere una variabile stringa.

Formalismo sintattico:

Esempio:

```
10 OPEN "PIPPO" AS # 1 LEN = 30
20 FIELD # 1, 10 AS NOME$, 20 AS VIA$
30 FOR I = 1 TO 5
40 GET # 1, I
50 PRINT NOME$, VIA$
60 NEXT I
```

Risultato:

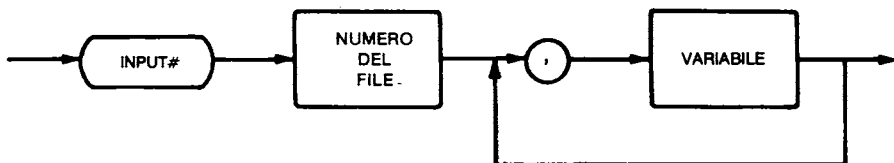
LA LINEA 10 apre un file "random" (ad accesso diretto) con numero di file 1.
LA LINEA 20 definisce la grandezza ed il nome dei campi del "buffer" (memoria di transito).
LA LINEA 30 innesca un ciclo da ripetersi 5 volte.
LA LINEA 40 porta i dati dall'unità esterna nel "buffer".
LA LINEA 50 stampa i campi letti.
LA LINEA 60 fa terminare il ciclo iniziato alla linea 30.

Note:
- NUMERO DEL FILE è un'espressione numerica i cui valori devono essere compresi fra 1 e 3 per IBM P.C., fra 1 e 15 per M20.
- LUNGHEZZA DEL CAMPO è una espressione numerica.
- NOME DEL CAMPO deve essere una variabile stringa.
- Bisogna evitare di usare altrove nel programma (per esempio in istruzioni LET o INPUT) gli stessi nomi di variabili utilizzati in una istruzione FIELD.

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

INPUT # (Istruzione)

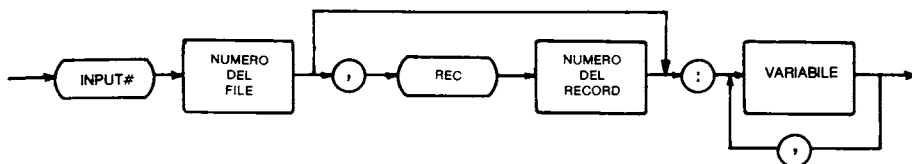
Legge i dati da un file sequenziale e li assegna a variabili di programma. Per TI 99/4A legge anche da file "relative" (ad accesso diretto).

Formalismo sintattico:

Esempio:

```
100 OPEN "PIPPO" FOR INPUT AS # 1
110 INPUT # 1, NOME$, VIA$
120 PRINT NOME$, VIA$
```

Risultato:

LA LINEA 100 apre il file sequenziale PIPPO$ con numero di file 1.
LA LINEA 110 legge 2 campi e li assegna alle variabili NOME$ e VIA$.
LA LINEA 120 stampa il contenuto dei campi letti.

Variante per: TI 99/4A

Note:

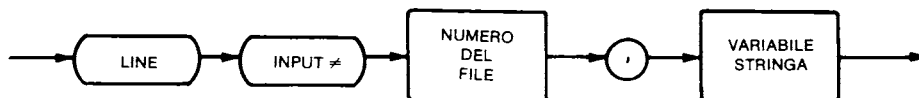
— NUMERO DEL FILE è un'espressione numerica i cui valori devono essere compresi fra 1 e 3 per IBM P.C., fra 1 e 15 per M20, fra 4 e 15 per ZX Spectrum, fra 1 e 127 per VIC 20 e CBM 64 e fra 0 e 255 per TI 99/4A.

— NUMERO DEL RECORD è un'espressione numerica.
— L'opzione REC si può usare solo con file "RELATIVE" ad accesso diretto.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

LINE INPUT # (Istruzione)

Legge un'intera linea (fino al carattere di ritorno a capo) da un file sequenziale e la assegna ad una variabile stringa.

Formalismo sintattico:

Esempio:

```
100 OPEN "PIPPO" FOR INPUT AS # 2
110 LINE INPUT # 2, A$
120 PRINT A$
```

Risultato:

LA LINEA 100 apre il file sequenziale PIPPO.
LA LINEA 110 legge una linea dal file e la assegna alla variabile A$.
LA LINEA 120 stampa sul video il contenuto della riga letta dal file.

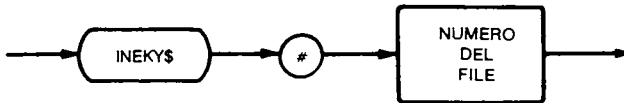
Nota:

— NUMERO DEL FILE è un'espressione numerica i cui valori devono essere compresi fra 1 e 3 per IBM P.C. e fra 1 e 15 per M20.

INKEY$ # (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Ritorna un solo carattere letto dal file specificato (il carattere successivo all'ultimo letto dal "buffer").

Formalismo sintattico:

Esempio:

```
100 OPEN # 4; "M"; 1; "PIPPO"
110 PRINT INKEY$ # 4
```

Risultato:

LINEA 100: apre il file PIPPO sul microdrive 1 assegnandogli il numero d'ordine 4.

LINEA 110: stampa a video il carattere letto dal "buffer" associato al file. Il carattere che viene letto è il primo non ancora letto disponibile nel buffer.

Note:

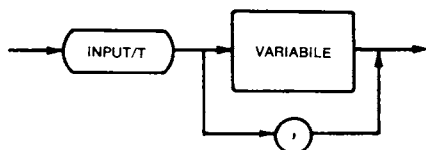
— Non funziona senza microdrive o interfaccia ZX1.

— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 4 e 15.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

INPUT/T (Istruzione)

È usata per immettere i dati da un file sequenziale su cassetta e passarli al programma.

Formalismo sintattico:

Esempio:

```
100 ROPEN "SCHEDINA"
110 DIM A$ (13)
120 FOR I = 1 TO 13
130 INPUT/T A$ (I)
140 PRINT A$ (I);
150 NEXT I
160 CLOSE
170 END
```

Risultato:

LINEA 100: apre il file chiamato "SCHEDINA" su cassetta e lo prepara alla lettura dei dati.

LINEA 110: dimensiona una matrice di 14 elementi che conterrà i dati prelevati dal file.

LINEA 120: inizia un ciclo di 13 ripetizioni.

LINEA 130: legge un valore dal file ad ogni ripetizione e lo memorizza in elementi della matrice A$.
L'elemento 0 della matrice non verrà utilizzato.

LINEA 140: stampa a video l'elemento di matrice appena riempito.

LINEA 150: fine del ciclo iniziato alla linea 120.

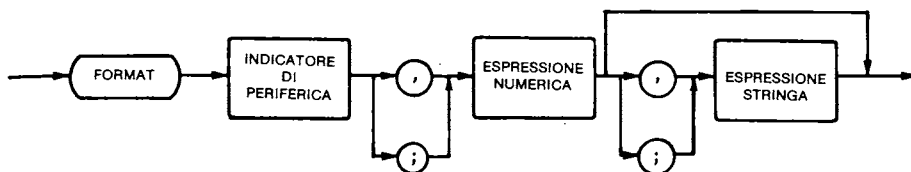
LINEA 160: chiusura del file "SCHEDINA". Dopo questa istruzione il file non è più interrogabile, ma bisogna riaprirlo di nuovo.

LINEA 170: fa terminare l'esecuzione del programma.

FORMAT (Comando)

	TI 99/4A
	VIC'20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Serve, se diretto al microdrive, a formattare la cartuccia. Assume significati diversi se diretto alla rete locale o all'interfaccia RS232.

Formalismo sintattico:

Esempio:
1) FORMAT "M"; 1; "PIPPO"
2) FORMAT A$; B; C$
3) FORMAT "N"; 3
4) FORMAT "T"; 4800.

Risultati:
Nell'esempio 1 formatta la cartuccia presente nel microdrive 1 e le assegna il nome PIPPO.
Nell'esempio 2 l'indicatore di periferica è contenuto nella variabile A$. Nella variabile B è contenuto il numero della periferica (da 1 a 8) e in C$ il nome della cartuccia (fino a 10 caratteri).
Nell'esempio 3 viene assegnato allo ZX Spectrum il numero di stazione 3.
Nell'esempio 4 viene modificata la velocità di trasmissione in 4800 Baud (Bit per secondo).

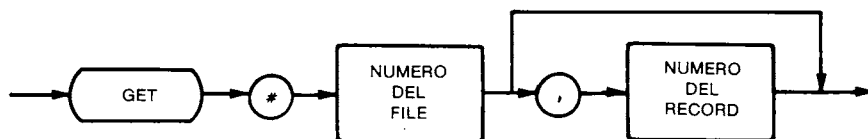
Note:
— Non funziona senza microdrive o interfaccia ZX1.
— L'indicatore di periferica è una espressione stringa che può assumere i valori:
- M, m per il microdrive
- N, n per la rete di servizio
- T, t oppure B, b per l'interfaccia RS232.

— L'espressione numerica identifica il numero della periferica indicata e può assumere valori compresi tra 1 e 8.
— L'espressione stringa è utilizzata solo per microdrive, e identifica il nome della cartuccia inserita (tra 1 e 10 caratteri).
— Può essere usato sia in modo immediato che in modo differito.

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

GET # (Istruzione)

Legge un record da un file ad accesso diretto.

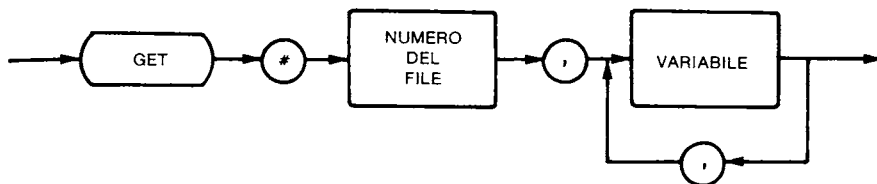
Formalismo sintattico:

Esempio:

```
10 OPEN "PIPPO" AS # 1
20 FIELD # 1, 10 AS NOME$, 20 AS VIA$
30 FOR I = 1 TO 5
40 GET # 1, I
50 PRINT NOME$, VIA$
60 NEXT I
```

Risultato:

LA LINEA 10 apre un file "random" (ad accesso diretto) con numero di file 1.
LA LINEA 20 definisce la grandezza ed il nome dei campi del "buffer" (memoria di transito).
LA LINEA 30 innesca un ciclo da ripetersi 5 volte.
LA LINEA 40 legge il record numerato della variabile I nel "buffer".
LA LINEA 50 stampa i campi letti.
LA LINEA 60 termina il ciclo iniziato alla linea 30.

Variante per: VIC 20

Variante per: CBM 64

Note:

— NUMERO DEL FILE è un'espressione numerica che può assumere valori da 1 a 3 per IBM P.C., da 1 a 15 per M20, da 1 a 127 per VIC 20 e CBM 64.

— NUMERO DEL RECORD è un'espressione numerica che può assumere valori compresi tra 1 e 32767.

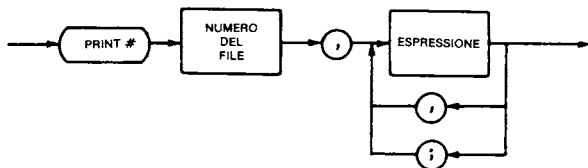
— L'istruzione GET, in IBM P.C., può essere usata anche per leggere un certo numero di byte, dato dal valore dell'espressione numerica NUMERO DEL RECORD, dalla memoria di transito di un file per le comunicazioni.

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

PRINT # (Istruzione)

Registra dati su un file sequenziale, usando lo stesso formato standard utilizzato da una istruzione PRINT.

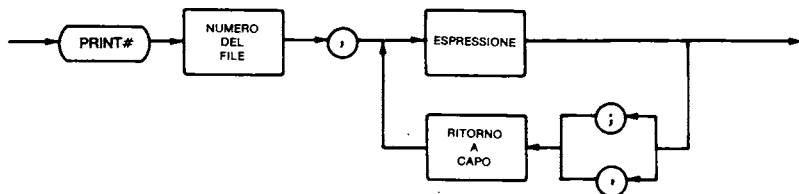
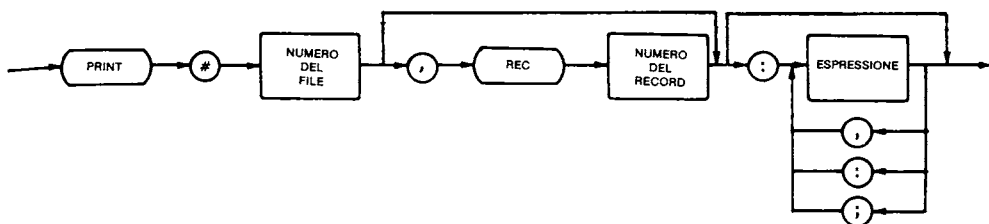
Nel TI 99/4A può scrivere su file "relative" (ad accesso diretto).

Formalismo sintattico:

Esempio:

```
100 OPEN "PIPPO" FOR OUTPUT AS # 1
110 A$ = "CAMERA"
120 B$ = "/201"
130 PRINT # 1, A$; B$
```

Risultato:

LA LINEA 100 apre il file sequenziale PIPPO per la scrittura.
LE LINEE 110 e 120 assegnano due valori stringa alle variabili A$ e B$.
LA LINEA 130 scrive sul file 1: CAMERA/201.

Variante per: VIC 20-CBM 64

Variante per: TI 99/4A

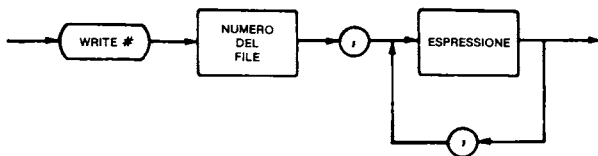
Note:

— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 3 per IBM P.C., fra 1 e 15 per M20, fra 4 e 15 per ZX Spectrum, fra 1 e 127 per VIC 20 e CBM 64 e fra 0 e 255 per TI 99/4A.
— NUMERO DEL RECORD è un'espressione numerica.
— L'opzione REC per TI 99/4A si può usare solo per file "RELATIVE", ad accesso diretto.
— Per RITORNO A CAPO vedere Appendice A.

WRITE # (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Registra dati su un file sequenziale usando lo stesso formato utilizzato dall'istruzione WRITE sul video. Le stringhe vengono delimitate dal carattere " (virgolette). Dopo la registrazione dell'ultimo dato della lista, il BASIC introduce un ritorno a capo (interlinea).

Formalismo sintattico:

Esempio:

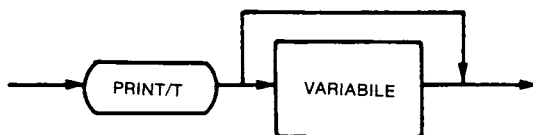
```
10 OPEN A$ FOR OUTPUT AS # 1
20 N = 5
30 WRITE # 1, N, N,
```

Risultato:
LA LINEA 10 apre un file sequenziale con numero di file 1.
LA LINEA 20 assegna alla variabile N il valore 5.
LA LINEA 30 scrive: 5,5 sul file.

Nota:
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 3 per IBM P.C. e fra 1 e 15 per M20.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

PRINT/T (Istruzione)

Scrive i dati sulla cassetta nello stesso formato in cui vengono visualizzati mediante l'istruzione PRINT.

Formalismo sintattico:

Esempio:

```
100 DIM A$ (13)
110 WOPEN "SCHEDINA"
120 FOR I = 1 TO 13
130 PRINT "RIGA"; I
140 GET B$: IF B$ = " " GOTO 140
150 IF (B$ = "1") + (B$ = "2") + (B$ = "X") GOTO 170
160 GOTO 140
170 A$ (I) = B$
180 PRINT/T A$ (I)
190 NEXT I
200 CLOSE
210 END
```

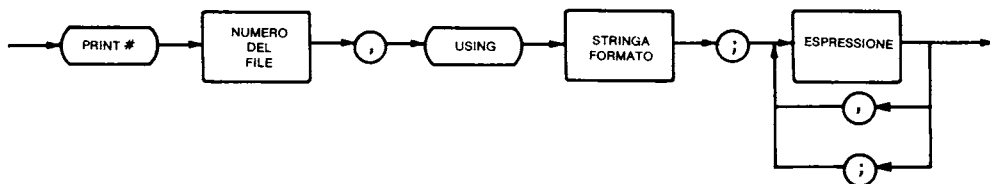
Risultato:

LINEA 100: dimensiona una matrice di 14 elementi che conterrà i dati da scrivere sul file.
LINEA 110: apre il file col nome "SCHEDINA" e lo prepara alla scrittura dei dati.
LINEA 120: inizia un ciclo di 13 ripetizioni.
LINEA 130: stampa la costante stringa e la variabile numerica specificate.
LINEA 140: attende un input da tastiera, che assegna alla variabile stringa B$.
LINEE 150÷160: controllo dei valori introdotti, che possono essere solo 1, 2 o X: in caso contrario il controllo viene spostato alla linea 140.
LINEA 170: il contenuto della variabile B$ viene assegnato ad un elemento della matrice A$. L'elemento 0 della matrice non verrà utilizzato.
LINEA 180: scrive il valore appena memorizzato in A$ sul file di output denominato "SCHEDINA".
LINEA 190: fine del ciclo iniziato alla linea 120.
LINEA 200: chiusura del file "SCHEDINA".
LINEA 210: fa terminare l'esecuzione del programma.

PRINT # USING (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Registra dati su un file sequenziale, usando lo stesso formato definito dall'utente utilizzato da una istruzione PRINT USING sul video.

Formalismo sintattico:

Esempio:

```
100 OPEN "PERSON" FOR OUTPUT AS # 1
110 A$ = "PIPPO" : B = 2000.50
120 PRINT # 1, USING "LO STIPENDIO DI & È # # # # # .# #";
    A$; B
```

Risultato:

LINEA 100: apre il file PERSON per la scrittura e gli assegna il numero d'ordine 1.
LINEA 110: assegna i valori specificati alle variabili A$ e B.
LINEA 120: scrive sul file 1 il seguente record:
LO STIPENDIO DI PIPPO È 2000.50.

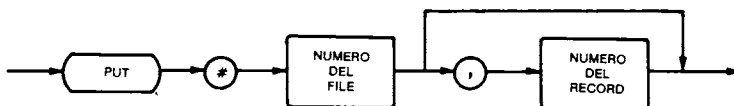
Note:

— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 3 per IBM P.C. e fra 1 e 15 per M20.
— I caratteri di formattazione utilizzabili all'interno della STRINGA FORMATO sono parecchi e diversificati nei vari dialetti. Rimandiamo all'istruzione PRINT USING e all'allegato A (alla voce STRINGA FORMATO) per la loro spiegazione.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

PUT # (Istruzione)

Registra su un file ad accesso diretto i dati prelevati dal buffer di I/O associato al file.

Formalismo sintattico:

Esempio:

```
10 OPEN "C : PIPPO" AS # 3
20 FIELD 3, 4 AS ANNO$, 4 AS TIPO
30 LSET ANNO$ = "1980" : LSET TIPO = "1010"
40 PUT # 3, 1
```

Risultato:

LA LINEA 10 apre un file ad accesso diretto di nome PIPPO sul drive C con numero di file 3.
LA LINEA 20 definisce il "buffer" di I/O (memoria di transito).
LA LINEA 30 assegna dei valori alle variabili ANNO$ e TIPO.
LA LINEA 40 scrive il record numero 1 nel file 3.

Note:
- NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 3 per IBM P.C. e fra 1 e 15 per M20.
- NUMERO DEL RECORD è un'espressione numerica che può assumere valori compresi fra 1 e 32767.

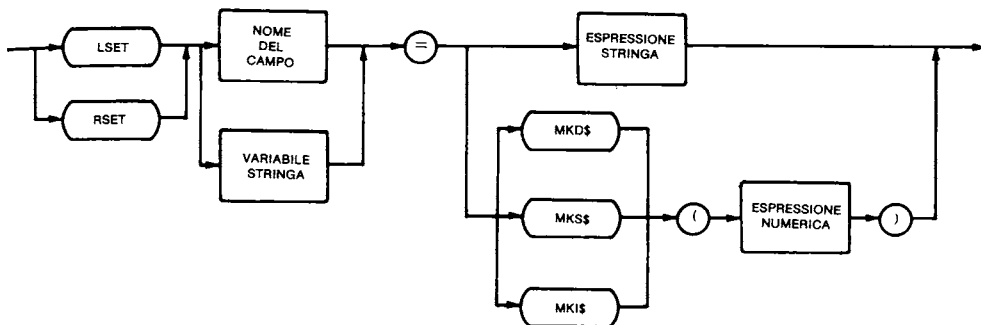
LSET/RSET (Istruzioni)

X	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

LSET memorizza un valore stringa, allineato a sinistra, in un campo del buffer ad accesso diretto oppure allinea a sinistra un valore stringa in una variabile stringa.

RSET è come LSET, solo che l'allineamento viene eseguito a destra anzichè a sinistra.

Entrambe preparano quindi il buffer per una istruzione PUT #.

Formalismo sintattico:

Esempio:

```
20 OPEN "PIPPO" AS # 2
30 FIELD # 2, 10 AS NOME$, 20 AS VIA$
40 LINE INPUT "NOME :"; N$
50 LINE INPUT "VIA :"; V$
60 LSET NOME$ = N$
70 LSET VIA$ = V$
80 PUT # 2, I%
```

Risultato:

LA LINEA 20 apre un file ad accesso diretto di nome PIPPO con numero d'ordine 2.

LA LINEA 30 definisce il "buffer" (memoria di transito).

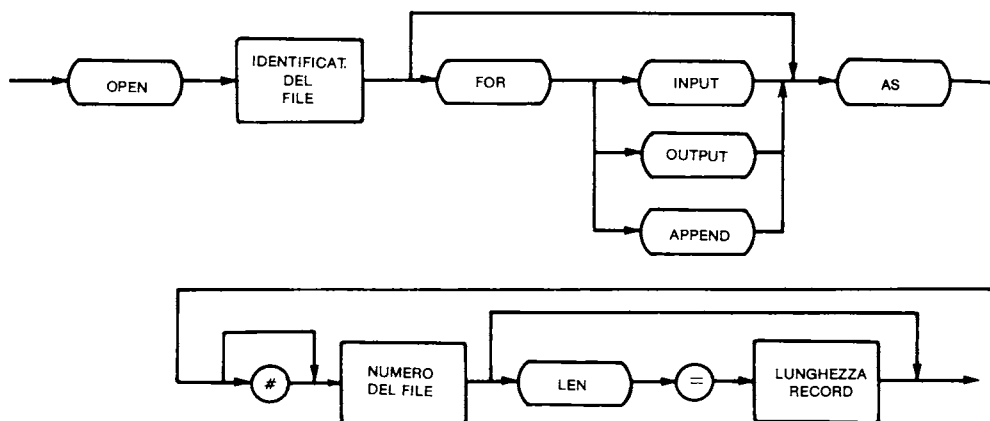
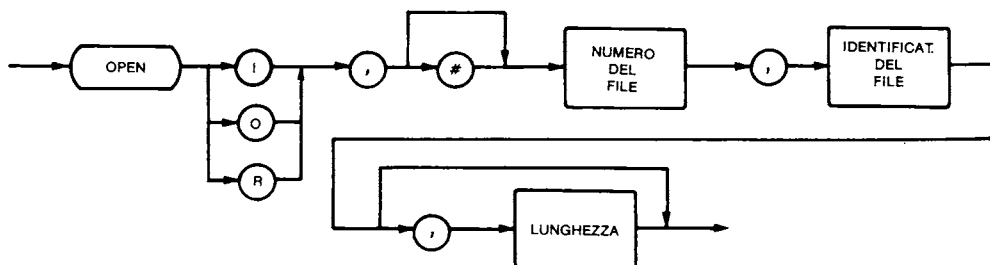
LE LINEE 40 e 50 assegnano due input, rispettivamente, alle variabili N$ e V$.
LE LINEE 60 e 70 assegnano ai campi del buffer il contenuto delle variabili N$ e V$ allineato a sinistra.
LA LINEA 80 scrive il contenuto del "buffer" sul file 2.

X	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	•OLIVETTI M20
	APPLE

OPEN (Istruzione)

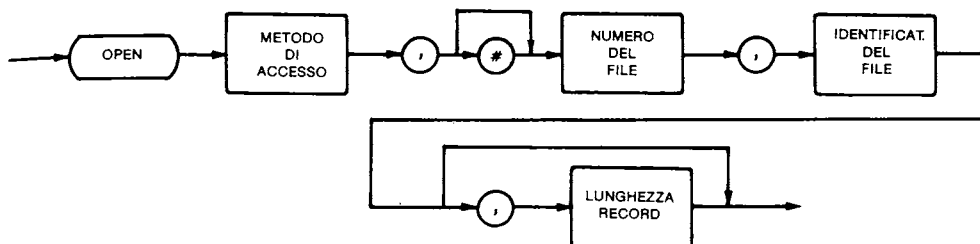
Apre un file dati consentendone l'accesso da programma.

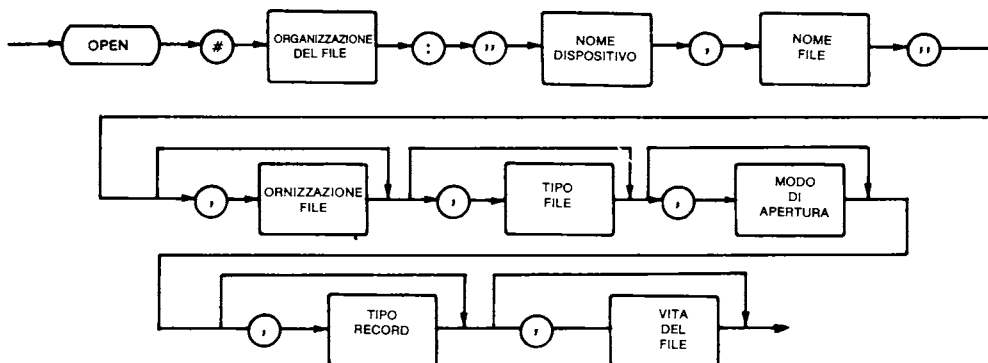
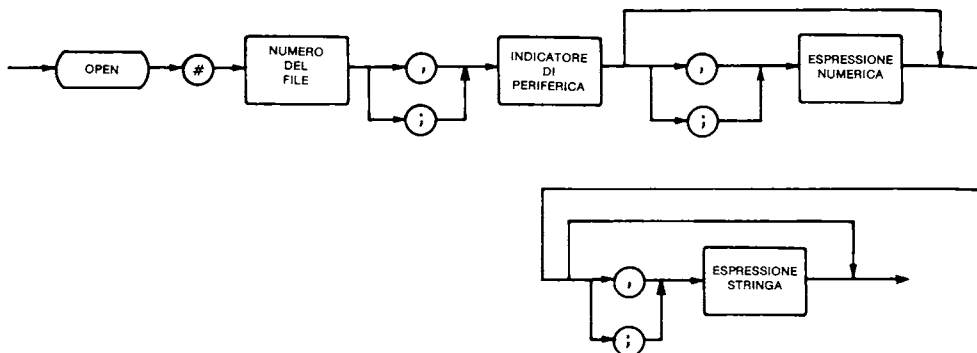
Formalismo sintattico per: IBM P.C.

formato 1

Esempi:

1) 10 OPEN "PIPPO" FOR OUTPUT AS # 2 LEN = 128
2) 10 OPEN "I", # 1, "B : PIPPO", 256

Risultati:
Nell'esempio 1 viene aperto un file in sola scrittura con numero di file 2 e lunghezza record 128.
Nell'esempio 2 viene aperto un file in sola lettura con numero di file 1 e lunghezza record 256.

Formalismo sintattico per: M20

Formalismo sintattico per: VIC 20 - CBM 64

Formalismo sintattico per: TI 99/4A

Formalismo sintattico per: ZX Spectrum.

Note per IBM P.C.:

— NUMERO DEL FILE è un'espressione numerica che può assumere valori da 1 al numero massimo di file consentiti, che all'atto dell'accensione è 3. È però possibile modificare il numero massimo di file.
— Per IDENTIFICATORE DEL FILE vedere l'Appendice A.
— LUNGHEZZA RECORD è un'espressione numerica che può assumere valori compresi fra 1 e 32767. Non è valida per i file sequenziali.

Note per Olivetti M20:

— Per METODO DI ACCESSO vedere l'allegato A.
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 15.
— Per IDENTIFICATORE DEL FILE vedere l'allegato A.
— LUNGHEZZA RECORD è un'espressione numerica che può assumere valori compresi tra 1 e 4096. Non è valida per i file sequenziali.

Note per VIC 20 e CBM 64:

— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 127.
— DISPOSITIVO è un'espressione numerica che può assumere i seguenti valori:

0	Tastiera
1	Registratore a cassetta
2	Modem
3	Schermo video
4 ÷ 5	Stampante
8 ÷ 11	Minidisco.

— INDIRIZZO SECONDARIO è un'espressione numerica il cui valore dipende da DISPOSITIVO e può variare da 0 a 15.
— Per IDENTIFICATORE DEL FILE vedere l'Appendice A.

Note per TI 99/4A:
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi tra 0 e 255.
— Per NOME DISPOSITIVO, ORGANIZZAZIONE FILE, TIPO FILE, MODO DI APERTURA e TIPO RECORD vedere l'Appendice A.
— NOME FILE è un'espressione stringa.
— VITA DEL FILE è un'espressione stringa. Se omessa, i file sono considerati PERMANENT.

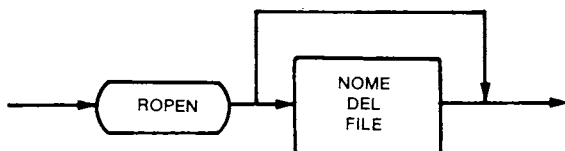
Note per ZX Spectrum:
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 4 e 15.
— INDICATORE DI PERIFERICA è un'espressione stringa che può assumere i valori:
- M, m per il microdrive
- N, n per la rete di servizio
- T, t oppure B, b per l'interfaccia RS232

— ESPRESSIONE NUMERICA può assumere valori compresi tra 1 e 8 e determina il numero della periferica.
— ESPRESSIONE STRINGA può essere lunga fino a 10 caratteri e contiene il nome del file che si vuole aprire. È utilizzata solo per microdrive.
— L'istruzione OPEN non funziona senza microdrive o interfaccia ZX 1.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

ROPEN (Istruzione)

Cerca il file di dati sequenziale su cassetta con il nome specificato ed apre tale file per preparare la lettura dei dati dello stesso.

Formalismo sintattico:

Esempio:

```
100 ROPEN "SCHEDINA"
110 DIM A$ (13)
120 FOR I = 1 TO 13
130 INPUT/T A$ (I)
140 PRINT A$ (I);
150 NEXT I
160 CLOSE
170 END
```

Risultato:

LINEA 100: apre il file chiamato "SCHEDINA" su cassetta e lo prepara alla lettura dei dati.
LINEA 110: dimensiona una matrice di 14 elementi che conterrà i dati prelevati dal file.
LINEA 120: inizia un ciclo di 13 ripetizioni.
LINEA 130: legge un valore dal file ad ogni ripetizione e lo memorizza in elementi della matrice A$. L'elemento 0 della matrice non verrà utilizzato.
LINEA 140: stampa l'elemento di matrice appena riempito.
LINEA 150: fine del ciclo iniziato alla linea 120.
LINEA 160: chiusura del file "SCHEDINA". Dopo questa istruzione il file non è più interrogabile, ma .bisogna riaprirlo di nuovo.
LINEA 170: fa terminare l'esecuzione del programma.

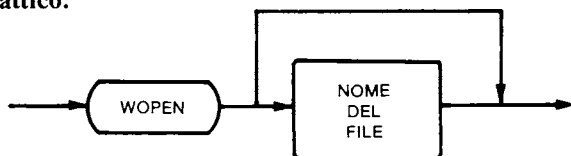
Note:

— NOME DEL FILE è un'espressione stringa.
— Il file che viene letto nell'esempio viene creato in un modo visibile nell'esempio annesso all'istruzione WOPEN.

WOPEN (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Apre un file di dati sulla cassetta prima di scrivervi i dati. Assegna anche un nome al file di dati.

Formalismo sintattico:

Esempio:

```
100 DIM A$ (13)
110 WOPEN "SCHEDINA"
120 FOR I = 1 TO 13
130 PRINT "RIGA" ; I
140 GET B$ : IF B$ = " " GOTO 140
150 IF (B$ = "1") + (B$ = "2") + (B$ = "X") GOTO 170
160 GOTO 140
170 A$ (I) = B$
180 PRINT/T A$ (I)
190 NEXT I
200 CLOSE
210 END
```

Risultato:

LINEA 100: dimensiona una matrice di 14 elementi che conterrà i dati da scrivere sul file.

LINEA 110: apre il file e lo prepara alla scrittura dei dati, assegnandogli contemporaneamente il nome "SCHEDINA".

LINEA 120: inizia un ciclo di 13 ripetizioni.

LINEA 130: stampa la costante stringa e la variabile numerica specificate.

LINEA 140: attende un input da tastiera, che assegna alla variabile stringa B$.

LINEE 150÷160: controllo dei valori introdotti, che possono valere solo 1, 2 o X: in caso contrario il controllo viene spostato alla linea 140.

LINEA 170: il contenuto della variabile B$ viene assegnato ad un elemento della matrice A$. L'elemento 0 della matrice non verrà utilizzato.

LINEA 180: scrive il valore appena memorizzato in A$ sul file di output denominato "SCHEDINA".

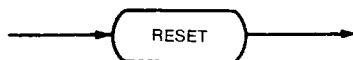
LINEA 190: fine del ciclo iniziato alla linea 120.
LINEA 200: chiusura del file "SCHEDINA".
LINEA 210: fa terminare l'esecuzione del programma.

Note:
— NOME DEL FILE è un'espressione stringa.
— Il file creato in questo esempio può essere utilizzato in un modo visibile nell'esempio annesso all'istruzione ROPEN.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

RESET (Comando)

Chiude tutti i file su minidisco e svuota la memoria di transito del sistema.

Formalismo sintattico:

Esempio:

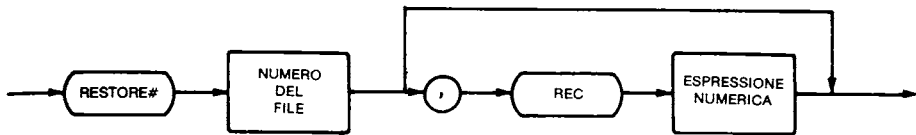
```
500 PRINT "INSERISCI L'ARCHIVIO NEL DRIVE B"
510 END
> RESET
```

Risultato:
LA LINEA 500 stampa il messaggio specificato.
LA LINEA 510 fa terminare l'esecuzione del programma.
Il comando RESET chiude tutti i file aperti e cancella il "buffer" (memoria di transito).

RESTORE # (Istruzione)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Riposiziona un file già aperto sul record iniziale oppure, se il file è di tipo RELATIVE, su un record specificato.

Formalismo sintattico:

Esempio:

```
 100 OPEN # 1 : "CS1", SEQUENTIAL, INTERNAL, INPUT, FIXED 80
 110 INPUT # 1 : A$, B$, C, D, E$
 120 .......
      :
1000 .......
1010 RESTORE # 1
1020 INPUT # 1 : A$, B$, C, D, E$
1030 CLOSE # 1
1040 END
```

Risultato:

LINEA 100: apre un file sequenziale per la lettura, con record di 80 caratteri, dalla cassetta 1, assegnandogli il numero di file 1.
LINEA 110: legge un record dal file, riempiendo le variabili specificate.
LINEE 120 ÷ 1000: resto del programma.
LINEA 1010: riposiziona il file sul record iniziale.
LINEA 1020: legge un record dal file, riempiendo le variabili specificate. Le variabili avranno un contenuto identico a quello che avevano dopo l'esecuzione della linea 110.
LINEA 1030: chiude il file.
LINEA 1040: fa terminare l'esecuzione del programma.

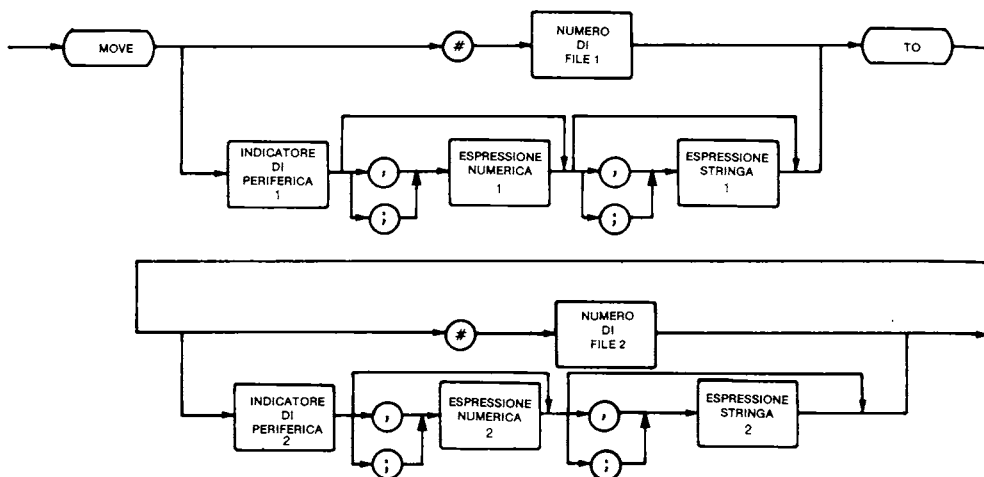
Note:

— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 0 e 255.
— L'opzione REC si può usare solo per file "RELATIVE", ad accesso diretto.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

MOVE (Comando)

Preleva i dati da un file sorgente e li invia ad un file di destinazione.

Formalismo sintattico:

Esempio:

1) MOVE # 5 TO # 4
2) MOVE "N"; 3 TO "M"; 1; "DATI MODEM"
3) MOVE #4 TO "N"; 15.

Risultati:
Nell'esempio 1 i dati, prelevati dal file 4, vengono inviati al file 5. Il file 4 dovrà essere stato aperto per la lettura e il file 5 per la scrittura.
Nell'esempio 2 i dati ricevuti dalla stazione 3 vengono inviati al file "DATI MODEM" sul microdrive 1.
Nell'esempio 3 i dati prelevati dal file 4 verranno inviati alla stazione 15. Il file 4 dovrà essere stato aperto per la lettura.

Note:
- NUMERO DI FILE (1 e 2) sono espressioni numeriche che possono assumere valori compresi tra 4 e 15.
- Per INDICATORE DI PERIFERICA (1 e 2), ESPRESSIONE NUMERICA (1 e 2), ESPRESSIONE STRINGA (1 e 2) vedere l'istruzione OPEN.
- Il comando MOVE può anche funzionare in modo differito, ma solo con microdrive o interfaccia ZX 1.

EOF (Funzione)

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Ritorna - 1 (cioè vero) se è stata raggiunta la fine di un file sequenziale, altrimenti ritorna 0.

Formalismo sintattico:

Esempio:

```
  10 OPEN "PIPPO" FOR INPUT AS # 1
  20 WHILE 1
  30 IF EOF (1) THEN 1000
  40 INPUT # 1, NOME$, VIA$
  50 PRINT NOME$, VIA$
  60 WEND
     :
1000 REM * SEGUITO *
```

Risultato:

LA LINEA 10 apre il file sequenziale PIPPO con numero di file 1.
LA LINEA 20 innesca un ciclo.
LA LINEA 30, se è verificata la condizione di fine file per il file PIPPO, sposta il controllo alla linea 1000.
LA LINEA 40 effettua la lettura di un record del file PIPPO.
LA LINEA 50 stampa a video i campi letti.
LA LINEA 60 ritorna alla 20 per chiudere il ciclo.

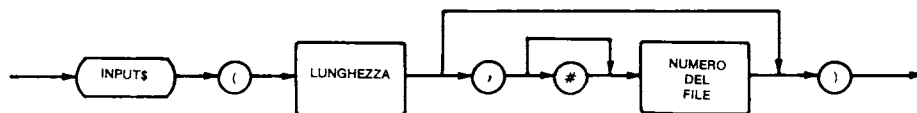
Nota:

— NUMERO DEL FILE è un'espressione numerica che può assumere valori da 1 a 3 per IBM P.C., da 1 a 15 per M20 e da 0 a 255 per TI 99/4A.

INPUT$ (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Calcola una stringa, di una lunghezza specificata, impostata da tastiera o letta da un file su disco.

Formalismo sintattico:

Esempio:

```
100 PRINT "DATA IN 6 CARATTERI"
110 D$ = INPUT$ (6)
```

Risultato:

LA LINEA 100 stampa la frase: DATA IN 6 CARATTERI.
LA LINEA 110 interrompe l'esecuzione del programma fino a quando sono stati digitati 6 caratteri. Tale stringa sarà assegnata alla variabile D$.

Note:
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi tra 1 e 3 per IBM P.C. e fra 1 e 15 per M20.
— LUNGHEZZA è un'espressione numerica che può assumere valori compresi fra 1 e 255.

LOC (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Nel caso di file sequenziali, ritorna il numero di settori letti o registrati su un file da quando è stato aperto.
Nel caso di file ad accesso diretto, fornisce il numero del record appena letto tramite un'istruzione GET# o appena registrato tramite un'istruzione PUT#.

Formalismo sintattico:

Esempio:

```
  10 OPEN "PIPPO" FOR INPUT AS # 1
  20 :
     :
 100 IF LOC (1) > 100 THEN GOTO 1000
     :
1000 REM *
```

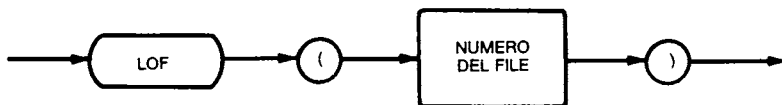
Risultato:
LA LINEA 10 apre il file sequenziale PIPPO con numero di file 1.
LA LINEA 100, se sono stati letti più di 100 record, sposta il controllo alla linea 1000.

Nota:
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 3 per IBM P.C. e fra 1 e 15 per M20.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

LOF (Funzione)

Ritorna il numero di byte allocati al file (lunghezza del file).

Formalismo sintattico:

Esempio:

```
100 OPEN "PIPPO" AS # 3 LEN = 128
110 PRINT LOF (3)
```

Risultato:
LA LINEA 100 apre un file ad accesso diretto di lunghezza 128 byte.
LA LINEA 110 stampa: 128.

Nota:
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 3.

CAPITOLO 14

COMANDI E ISTRUZIONI PARTICOLARI

ELENCO COMANDI E ISTRUZIONI

CHAIN
CMD
CALL JOYST
COM
COMMON
COPY
DEF KEY
KEY
KEY (n)
DEF SEG
DELETE
ERASE
KILL
FAST
CAT
FILES
CALL FILES
LIMIT
DEF USR
HIMEM :
LOMEM :
LLIST
LIST/P
MERGE
MOTOR
NAME
NULL
ON COM
ON KEY
ON PEN
ON STRIG
OPEN "COM
IN #
OUT
PR #
PEN
LPRINT
PRINT/P
LPRINT USING
PRINT/P USING
SLOW
STRIG
STRIG (n)
SWAP
USR
CALL
EXEC
WAIT
PAUSE
WHILE
WEND
SCROLL

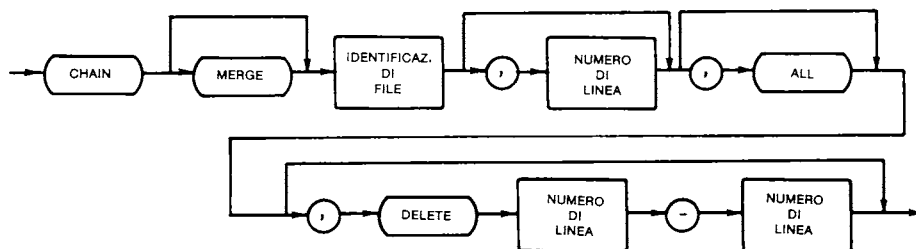
ELENCO FUNZIONI

JOY
STICK
PDL
IN
INP
SYS
LPOS
VARPTR
VARPTR$

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

CHAIN (Comando)

Consente di concatenare il programma specificato a quello attualmente in memoria, consentendo il trasferimento delle variabili. Lascia i file dati aperti e mantiene l'attuale impostazione di OPTION BASE (se si usa l'opzione MERGE).

Formalismo sintattico:

Esempio:
1) CHAIN "A:PROG1", 1010,ALL
2) CHAIN MERGE "B:PROG2", 100,ALL,DELETE 100-200

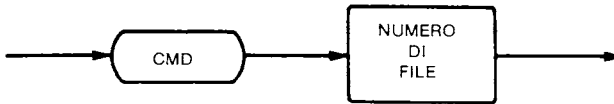
Risultato:
Nell'esempio 1 viene trasferito il controllo al programma PROG1, alla riga 1010, e tutte le variabili del programma corrente vengono passate a quello chiamato.
Nell'esempio 2 il programma PROG2 viene unito a quello in memoria, partendo dalla linea 100, vengono passate tutte le variabili e vengono cancellate le linee dalla 100 alla 200.

Note:
— può essere usato sia in modo immediato che in modo differito.
— NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65.535.
— per IDENTIFICATORE DI FILE vedere l'appendice A.
— il programma concatenato cancella il precedente (tranne, eventualmente, le variabili) e viene automaticamente messo in esecuzione. Per ritornare al programma precedente, quello che lo ha chiamato, bisogna ridare un'altra CHAIN.

CMD (Istruzione)

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

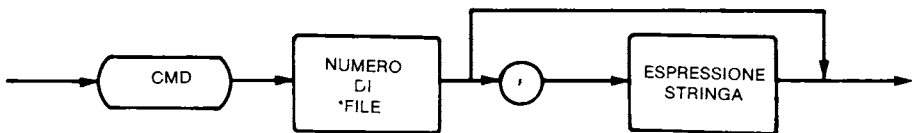
Invia tutti i dati, invece che a video, su una altra unità specificata.

Formalismo sintattico:

Esempio:

```
10 OPEN 2, 4
20 CMD 2
30 LIST
40 PRINT # 2
50 CLOSE 2
```

Risultato:

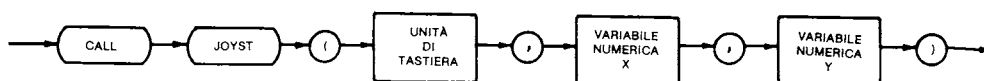
LINEA 10: apre un file sulla periferica 4 (stampante), a cui viene assegnato il numero d'ordine 2.
LINEA 20: commuta l'output dal video alla periferica precedentemente aperta.
LINEA 30: lista il programma attualmente in memoria sulla stampante anziché sul video.
LINEA 40: stampa un ritorno a capo e disabilita la stampante.
LINEA 50: chiude il file.

Variante per: CBM 64

Nota:
— NUMERO DI FILE è un'espressione numerica che può assumere valori compresi fra 1 e 127.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CALL JOYST (Istruzione)

Il sottoprogramma JOYST consente, nel TI 99/4A, di inviare informazioni sulla posizione del Joystick dei controlli a distanza.

Formalismo sintattico:

Esempio:

```
100 CALL CLEAR
110 CALL CHAR (65, "1898FF3D3C3CE404")
120 CALL JOYST (2, X, Y)
130 R = Y * 1.6 + 12.2
140 C = X * 2.2 + 16.6
150 CALL HCHAR (R, C, 65)
160 GOTO 120
```

Risultato:

LINEA 100: pulisce il video.

LINEA 110: genera il carattere "omino" (vedere "generazione caratteri" e l'istruzione CALL CHAR).

LINEA 120: assegna alle variabili X e Y le informazioni sulla posizione del Joystick 2.

LINEE 130÷140: calcolano un valore di R (riga) e C (colonna) che dipenda dalle variabili X e Y.

LINEA 150: stampa il carattere "omino" alla riga R e colonna C. Poiché tali variabili dipendono da X e Y, l'omino si sposterà sullo schermo in corrispondenza ai movimenti del Joystick 2.

LINEA 160: riporta il controllo alla linea 120.

Note:

— UNITA' DI TASTIERA è un'espressione numerica che può assumere i seguenti valori:

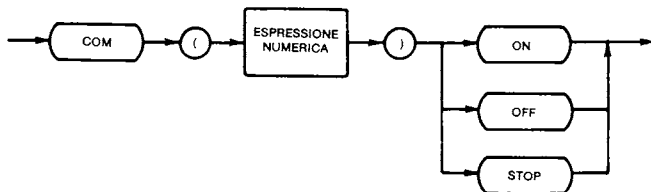
1 = controllo a distanza 1
2 = controllo a distanza 2
3, 4, 5 = modi particolari per la tastiera della console

— X e Y sono variabili numeriche in cui verrà messo un valore intero pari a −4, 0 o +4, in dipendenza della posizione del joystick.

COM (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Abilita o disabilita l'adattatore per le comunicazioni specificato, all'intercettazione dell'attività di comunicazione.

Formalismo sintattico:

Esempi:
1) COM (1) ON
2) COM (2) OFF
3) COM (1) STOP

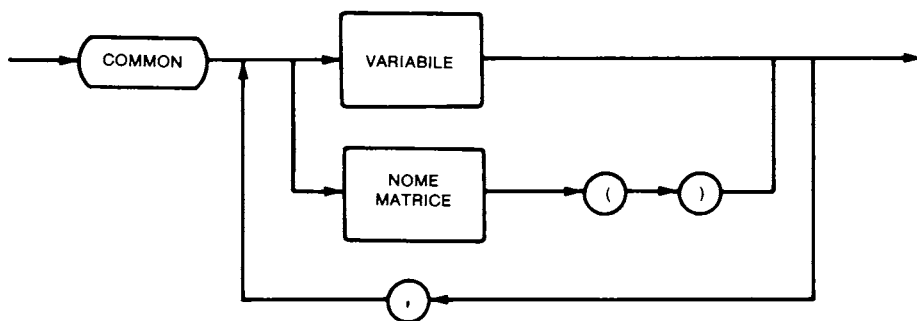
Risultato:
Nell'esempio 1 viene abilitato l'adattatore 1 per le comunicazioni.
Nell'esempio 2 viene disabilitato l'adattatore 2 per le comunicazioni.
Nell'esempio 3 viene spento l'adattatore 1 per le comunicazioni.

Note:
— deve essere sempre eseguita una istruzione COM (n) ON prima di una ON COM (n), se si vogliono intercettare i dati in arrivo sul canale delle comunicazioni.
— ESPRESSIONE NUMERICA può assumere i valori 1 e 2.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

COMMON (Istruzione)

Definisce un'area comune tra programmi concatenati che non viene ricoperta e permette quindi il trasferimento delle variabili.

Formalismo sintattico:

Esempio:

```
 90 REM * PROGRAMMA MAIN *
100 COMMON A, B, X, Y, Z
110 A = 10 : B = 20 : X = 1 : Y = 2 : Z = 3
120 CHAIN "PROG 1"

200 REM * PROGRAMMA PROG 1 *
210 COMMON A1, A2, A3, A4, A5
220 PRINT A1, A2, A3, A4, A5
230 END
```

Risultato:

Il programma principale assegna dei valori alle variabili A, B, X, Y, Z (linea 110). Alla linea 120 richiama il programma PROG1 che stampa: 10 20 1 2 3

Infatti le variabili, anche se assumono un nome diverso nei due programmi, vengono considerate in comune, e non vengono modificate dal comando CHAIN.

Note:

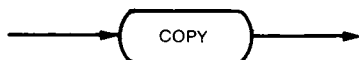
— Se è necessario passare al programma concatenato tutte le variabili, non usare COMMON ma l'opzione ALL del comando CHAIN.

— Nel programma concatenato non è necessario dimensionare le matrici in comune.

COPY (Comando)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Manda sulla stampante, se collegata, una copia dello schermo video (Hard-copy).

Formalismo sintattico:

Esempio:

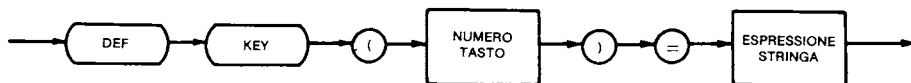
```
> LIST
10 REM * PROGRAMMA TEST *
20 LET A = 2
30 LET B = 3
40 LET C = A * B
50 END


> COPY
```

Risultato:
Il comando LIST stampa sul video il programma attualmente presente in memoria, mentre il comando COPY invia il contenuto dello schermo video, e quindi il listato del programma, sulla stampante.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

DEF KEY (Istruzione)

Permette di assegnare ai tasti funzionali programmabili la stringa voluta.

Formalismo sintattico:

Esempio:

```
10 DEF KEY (1) = "CHAIN"
20 DEF KEY (2) = "CURSOR X, Y"
30 DEF KEY (10) = "RUN" + CHR$ (13)
```

Risultato:

LINEA 10: assegna al tasto funzionale 1 la stringa specificata, e cioè il comando CHAIN.

LINEA 20: definisce il tasto funzionale 2 come la istruzione CURSOR X,Y.

LINEA 30: definisce il tasto funzionale 10 come RUN CR. Infatti 13 è il codice ASCII del tasto CR (cioè l'ENTER).

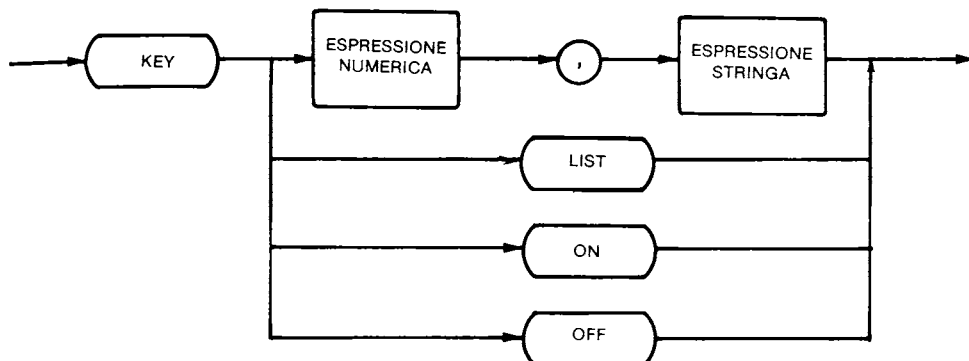
Note:

— Non è possibile scrivere un'altra istruzione sulla stessa riga in cui è presente una DEF KEY.
— L'esecuzione di una DEF KEY per un tasto funzionale annulla il precedente valore del tasto stesso.
— NUMERO TASTO è un'espressione numerica che può assumere valori compresi fra 1 e 10.
— ESPRESSIONE STRINGA può essere lunga fino a 15 caratteri.

KEY (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

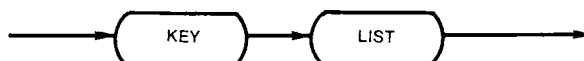
Imposta o visualizza i tasti temporanei (o tasti funzionali).

Formalismo sintattico:

Esempio:

```
10 KEY 1, "CHAIN"
20 KEY 10, "RUN" + CHR$ (13)
30 KEY LIST
40 KEY ON
```

Risultato:

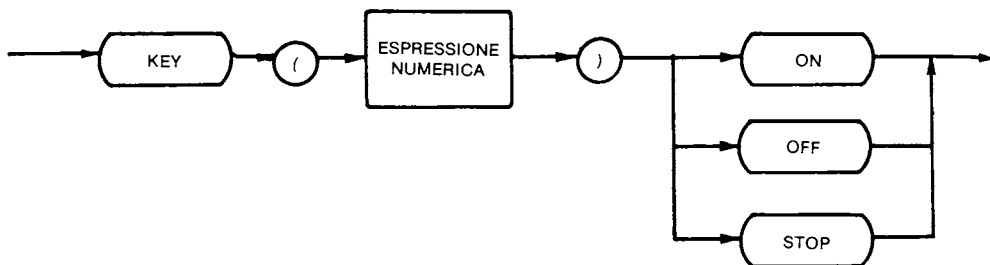
LINEA 10: imposta il tasto funzionale 1 col comando CHAIN.
LINEA 20: imposta il tasto funzionale 10 come RUN ENTER. Infatti 13 è il codice ASCII del tasto ENTER.
LINEA 30: lista a video il valore impostato ai dieci tasti funzionali.
LINEA 40: utilizza l'ultima riga dello schermo per visualizzare il valore impostato a 5 (o 10) tasti funzionali. 5 tasti vengono visualizzati con 40 colonne, 10 con 80 colonne. Vengono stampati solo i primi 6 caratteri del valore del tasto.

Variazione per: MZ 700

Note:
— ESPRESSIONE NUMERICA indica il tasto funzionale e può assumere valori da 1 a 10.
— ESPRESSIONE STRINGA può essere lunga fino a 15 caratteri.
— KEY OFF toglie la visualizzazione dei tasti funzionali sull'ultima riga del video.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

KEY (n) (Istruzione)

Attiva o disattiva l'intercettazione del tasto funzione specificato, in un programma BASIC.

Formalismo sintattico:

Esempio:

```
  100 KEY (1) ON
  110 KEY (10) ON
      :
  500 ON KEY (1) GOSUB 1000
  510 ON KEY (10) GOSUB 2000
      :
 1000 REM * SUBROUTINE F1 *
 2000 REM * SUBROUTINE F10 *
```

Risultato:

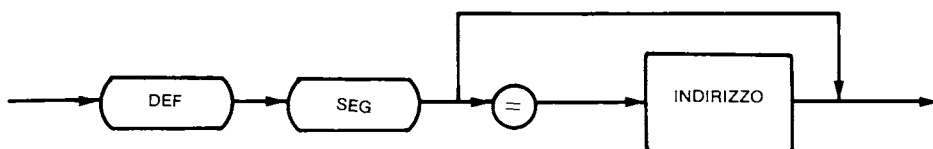
LINEE 10÷20: attivano l'intercettazione dei tasti funzione 1 e 10.
LINEA 500: sposta il controllo dell'esecuzione alla linea 1000 se è stato premuto il tasto funzione 1.
LINEA 510: sposta il controllo dell'esecuzione alla linea 2000 se è stato premuto il tasto funzione 10.

Note:
— deve sempre essere eseguita una istruzione KEY (n) ON prima di una ON KEY (n), se si vogliono intercettare i tasti funzionali.
— ESPRESSIONE NUMERICA può assumere valori compresi tra 1 e 14 col seguente significato:
1÷10 tasti funzionali (F1÷F10)
11 cursore verso l'alto
12 cursore a sinistra
13 cursore a destra
14 cursore in basso

DEF SEG (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Nell'I.B.M. P.C. definisce l'attuale "segmento" di memoria. Una successiva istruzione BLOAD, BSAVE, CALL, PEEK, POKE o USR definisce il reale indirizzo fisico dell'operazione come posizione relativa in questo segmento.

Formalismo sintattico:

Esempio:

```
  10 DEF SEG = & HB800
     :
1000 DEF SEG
```

Risultato:
LA LINEA 10 definisce l'indirizzo assoluto B800 esadecimale (buffer dello schermo per l'adattatore di controllo del colore) come indirizzo di inizio dell'attuale segmento di memoria.
LA LINEA 1000 ripristina il segmento iniziale, cioè quello dei dati del BASIC.

Note:
— INDIRIZZO è un'espressione numerica che può assumere valori compresi fra 0 e 65535 (oppure &H0000 e &HFFFF).

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

DELETE (Comando)

Serve per cancellare file di programmi o dati memorizzati su un'unità a disco.

Formalismo sintattico:

Esempio: **in modo immediato:**

1) DELETE "DSK1. FILE1"
2) DELETE "DSK2.SCHEDINA"

in modo differito:

```
100 CALL CLEAR
110 OPEN # 1 : "DSK2 . SCHEDINA"
120 GOSUB 1000
130 CLOSE # 1
140 DELETE "DSK2 . SCHEDINA"
150 END
    :
1000 REM * LETTURA E TRATTAMENTO FILE *
    :
1990 RETURN
```

Risultato: in modo immediato:

1) cancella il file FILE1 sull'unità a disco 1.
2) cancella il file SCHEDINA sull'unità a disco 2.

in modo differito:

alla linea 100 pulisce il video. Alla linea 110 apre il file SCHEDINA sull'unità a disco 2. Alla linea 120 richiama la subroutine che presiede alla lettura e al trattamento delle informazioni presenti sul file. Alla linea 130 chiude il file SCHEDINA e alla linea 140 lo cancella dall'unità a dischi 2. La linea 150 fa terminare l'esecuzione del programma.

Note:

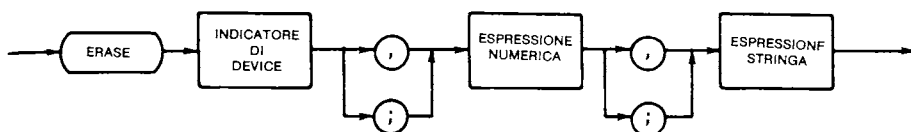
— nell'esempio visto, (in modo differito) la cancellazione poteva anche essere effettuata nel seguente modo: CLOSE # 1 : DELETE

— non è possibile cancellare file protetti
— vedere anche appendice A (per NOME DISPOSITIVO)

ERASE (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Cancella il file specificato nel microdrive specificato.

Formalismo sintattico:

Esempio:
1) ERASE "M"; 2; "PIPPO"
2) ERASE A$; B; C$

Risultato:
nell'esempio 1 viene cancellato il file PIPPO dalla cartuccia inserita nel microdrive 2.
nell'esempio 2 il nome del file cancellato è contenuto in C$, il numero del microdrive è contenuto in B, mentre A$ conterrà "M" o "m".

Note:
— INDICATORE DI DEVICE è una espressione stringa che può assumere solo i due valori "M" o "m".
— ESPRESSIONE NUMERICA specifica il numero del microdrive e può assumere valori compresi fra 1 e 8.
— ESPRESSIONE STRINGA specifica il nome del file da cancellare, e può essere

lunga fino a 10 caratteri.
— il comando ERASE funziona anche in modo differito, ma non funziona senza microdrive o interfaccia ZX1.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

KILL (Comando)

Cancella un file programma o un file dati registrato su disco.

Formalismo sintattico:

Esempio:
KILL "A:PROG1"

Risultato:
Cancella il file PROG1 sul drive A.

Note:
— funziona anche in modo differito.
— per IDENTIFICATORE DI FILE vedere appendice A (ricordarsi che per UNITA' è valido solo minidisco).

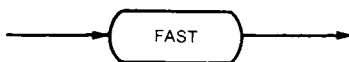
FAST (Comando)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Permette di far funzionare lo ZX 81 con la velocità maggiore. Tale velocità è indicata per eseguire calcoli lunghi.

Formalismo sintattico:

Esempio:

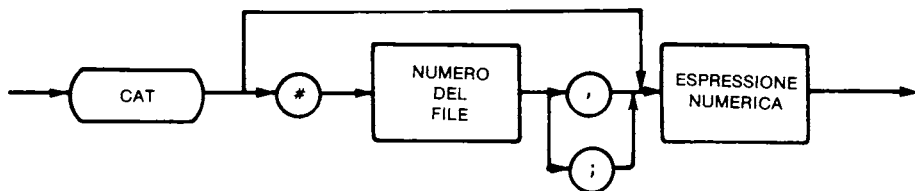
```
100 FAST
110 FOR R = 1 TO 100
120 LET C = 6.28 * R
130 LET A = 3.14 * R ** 2
140 LET V = (3.14 * 4/3) * R ** 3
150 PRINT C, A, V
160 NEXT R
170 SLOW
```

Risultato:

LINEA 100: imposta la velocità di elaborazione maggiore.
LINEA 110: inizia un ciclo da ripetersi 100 volte.
LINEE 120÷140: ad ogni ripetizione calcolano, rispettivamente, il valore della conferenza, dell'area e del volume della sfera di raggio R (da 1 a 100).
LINEA 150: stampa i valori calcolati.
LINEA 160: termine del ciclo iniziato alla 110.
LINEA 170: terminato il pezzo di programma contenente parecchi calcoli, ripristina la bassa velocità di esecuzione.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CAT (Comando)

Produce la lista dei primi 50 file non protetti presenti sulla cartuccia inserita nel microdrive specificato dall'espressione numerica. Viene anche stampata la quantità di spazio ancora disponibile sulla cartuccia stessa.

Formalismo sintattico:

Esempio:
1) CAT 2
2) CAT # 4; 1
3) CAT # A; B

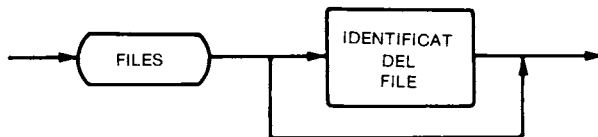
Risultato:
Nell'esempio 1 viene prodotta la lista dei file non protetti presenti sulla cartuccia inserita nel microdrive 2, che viene inviata al video.
Nell'esempio 2 viene prodotta la lista dei file non protetti presenti sulla cartuccia inserita nel microdrive 1, che viene inviata al file numero 4.
Nell'esempio 3 il numero del microdrive interessato è memorizzato in B, il numero del file a cui inviare la lista è presente in A.

Note:
— La lista viene normalmente inviata a video, a meno che non sia presente l'opzione # NUMERO DEL FILE, in cui viene specificato il numero del file a cui inviare la lista stessa.
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 0 e 15.
— ESPRESSIONE NUMERICA può assumere valori compresi fra 1 e 8.
— Il comando CAT funziona anche in modo differito, ma non funziona senza microdrive o interaccia ZX1.

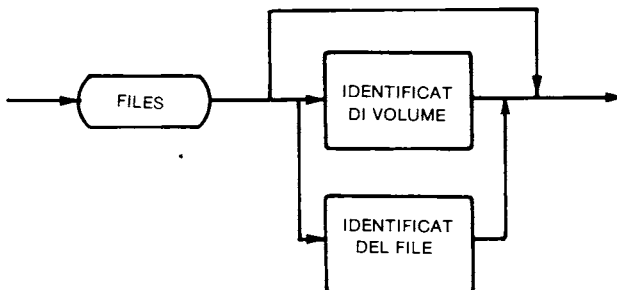
FILES (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Lista i file registrati sul disco specificato.

Formalismo sintattico per: I.B.M. P.C.

Esempio:
1) FILES
2) FILES “A * . COB”
3) FILES “A * . *”

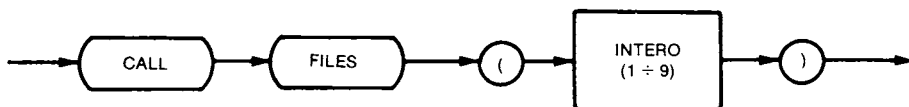
Risultato:
Nell’esempio 1 lista tutti i file del minidisco che il sistema operativo assume per difetto.
Nell’esempio 2 lista i file del minidisco A che hanno un nome qualsiasi ma con estensione uguale a .COB
Nell’esempio 3 lista tutti i file presenti sul minidisco A.

Formalismo sintattico per: M 20

Note:
— per IDENTIFICATORE DEL FILE e IDENTIFICATORE DEL VOLUME vedere l’appendice A.

X	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

CALL FILES (Comando)

Specifica il numero di file che possono venir aperti contemporaneamente. Se non si utilizza questo comando, il numero massimo consentito è 3. Subito dopo aver digitato questo comando deve venir necessariamente digitato NEW, altrimenti si possono ottenere risultati imprevedibili, compresa l'eventuale perdita delle informazioni registrate sui file che vengono aperti.

Formalismo sintattico:

Esempio:
>CALL FILES (4)   ENTER
>NEW   ENTER

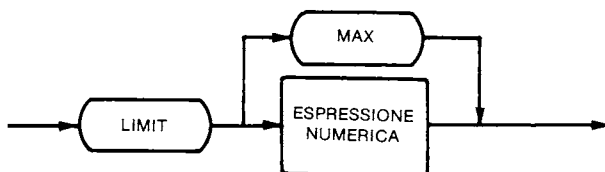
Risultato:
Specifica che, nel programma che verrà eseguito successivamente, è possibile gestire contemporaneamente fino a 4 file.

Note:
— CALL FILES può essere usato solo come comando in modo immediato.

LIMIT (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Limita l'area di memoria che può essere usata dall'interprete BASIC. L'area, dall'indirizzo seguente il valore dell'espressione numerica specificata fino all'indirizzo $FEFF, può essere usata per programmi in linguaggio macchina o dati speciali.

Formalismo sintattico:

Esempio:

1)
```
10 LIMIT 40959
20 LOAD "ORGANO"
30 USR (40960)
```

2)
```
10 LIMIT $9FFF
20 LOAD "ORGANO"
30 USR ($A000)
```

Risultato: i due esempi svolgono la stessa funzione, solo che nel primo caso vengono usate costanti decimali, nel secondo costanti esadecimali.

LINEA 10: limita l'area di memoria a disposizione del BASIC fino all'indirizzo 40959 ($9FFF). Ciò vuol dire che l'area di memoria dall'indirizzo 40960 ($A000) all'indirizzo 65279 ($FEFF) può essere utilizzata per registrare programmi in linguaggio macchina.

LINEA 20: carica il programma in linguaggio macchina ORGANO, che ha indirizzo di caricamento e di esecuzione uguale a 40960 ($A000).

LINEA 30: manda in esecuzione il programma in linguaggio macchina ORGANO.

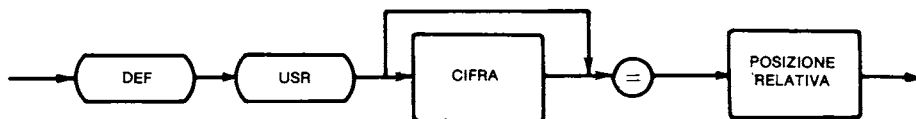
Note:

— LIMIT MAX viene utilizzata per annullare il limite che era stato definito in precedenza.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

DEF USR (Istruzione)

Specifica l'indirizzo di inizio di un sottoprogramma in linguaggio macchina. Questa istruzione deve essere eseguita prima che il sottoprogramma venga richiamato dalla funzione USR.

Formalismo sintattico:

Esempio:

```
10 DEF SEG = 2000
20 DEF USR0 = 20000
30 A = USR0 (B)
```

Risultato:

LINEA 10: definisce il segmento di memoria.

LINEA 20: definisce l'indirizzo di inizio del sottoprogramma in linguaggio macchina, con numero d'ordine 0, come 20.000. L'indirizzo reale di inizio del sottoprogramma 0 sarà all'indirizzo 22.000.

LINEA 30: richiama il sottoprogramma 0 in linguaggio macchina, che inizia all'indirizzo 22.000, passando come argomento la variabile B.

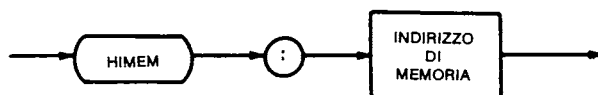
Note:

— POSIZIONE RELATIVA è un'espressione numerica che può assumere valori compresi fra 0 e 65535 (oppure da &H0000 a &HFFFF). Deve essere aggiunto al valore del segmento attuale se si vuole avere il vero indirizzo reale di inizio del sottoprogramma.

HIMEM: (Comando -istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

In APPLE II definisce l'indirizzo della posizione più elevata di memoria disponibile in un programma BASIC, comprese le variabili. Viene usato per proteggere l'area di memoria al di sopra di esso per i dati, i grafici o per le routine in linguaggio macchina.

Formalismo sintattico:

Esempio:

```
10 A = 16384
20 HIMEM : A
30 LOMEM : 2000
```

Risultato:

LA LINEA 10 assegna alla variabile A il valore 16384.
LA LINEA 20 definisce il massimo indirizzo per un programma BASIC come 16384.
LA LINEA 30 definisce che l'indirizzo 2000 è il più basso disponibile per un programma BASIC.

Note:

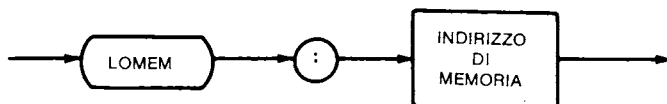
— INDIRIZZO DI MEMORIA è un'espressione numerica che può assumere valori compresi tra −65535 e 65535. I valori negativi vengono trattati come positivi.
— il valore presente in HIMEM: dovrà essere superiore a quello presente in LOMEM:

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

LOMEM: (Comando - istruzione)

In APPLE II definisce l'indirizzo della posizione di memoria più bassa disponibile per un programma BASIC.

Questo è solitamente l'indirizzo della posizione di memoria di inizio per la prima variabile BASIC. Questo comando consente la protezione delle variabili dai grafici ad alta risoluzione.

Formalismo sintattico:

Esempio:

```
10 A = PEEK (200)
20 LOMEM : 600 + A
30 HIMEM : 16384
```

Risultato:

LA LINEA 10 assegna alla variabile A il contenuto dell'indirizzo 200.

LA LINEA 20 definisce che l'indirizzo determinato dal valore dell'espressione numerica specificata, è il più basso disponibile per un programma BASIC.

LA LINEA 30 definisce che l'indirizzo 16384 è il più alto disponibile per un programma BASIC.

Note:

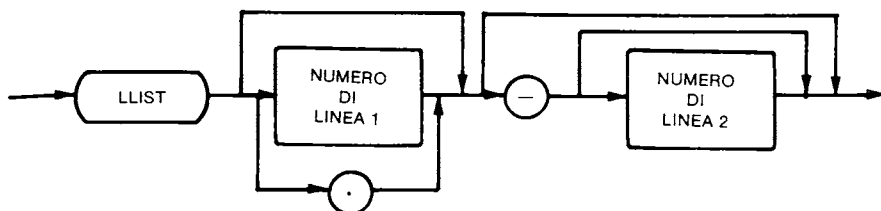
— INDIRIZZO DI MEMORIA è un'espressione numerica che può assumere valori compresi fra −65535 e 65535. I valori negativi vengono trattati come positivi.

— il valore presente in LOMEM: dovrà essere inferiore a quello presente in HIMEM:

LLIST (Comando)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Lista sulla stampante le linee di programma specificate.

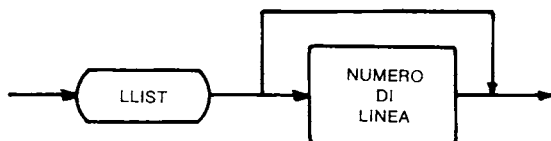
Formalismo sintattico:

Esempio: in modo immediato:

1) LLIST 10
2) LLIST 10 – 50
3) LLIST – 50
4) LLIST 50 –
5) LLIST

in modo differito:

6) 10 INPUT X : IF X = 1 THEN LLIST 10 – 100

Risultato:
1) Lista la linea 10 sulla stampante.
2) Lista tutte le linee del programma dalla 10 alla 50 incluse.
3) Lista su stampante tutte le linee dall'inizio del programma fino alla 50 inclusa.
4) Lista su stampante tutte le linee dalla 50 alla fine del programma.
5) Lista tutto il programma sulla stampante.
6) Lista su stampante tutte le linee dalla 10 alla 100 incluse se il valore digitato in risposta all'istruzione INPUT alla linea 10 è uguale a 1.

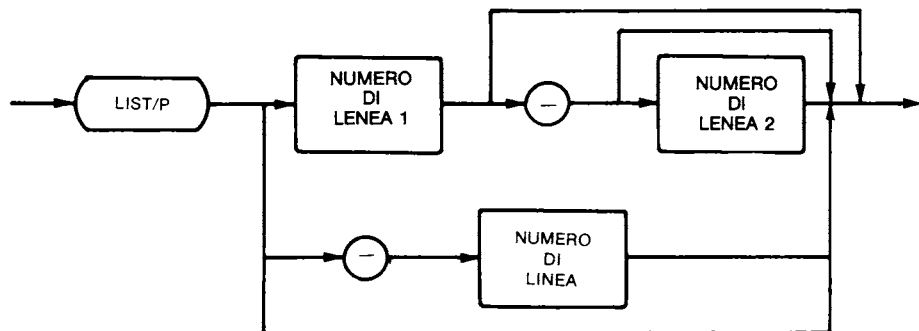
Variante per: ZX 81 - ZX Spectrum

Note:

— NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65.535. Il numero di linea 1 deve essere inferiore o uguale al numero di linea 2.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

LIST/P (Comando)

Stampa il programma, o una parte di esso, sul plotter ("modo "Testo").

Formalismo sintattico:

Esempi: In modo immediato:

1. LIST/P 10

lista sulla stampante la linea 10

2. LIST/P 10-50

Lista sulla stampante tutte le linee dalla 10 alla 50 incluse.

3. LIST/P -50

Lista sulla stampante tutte le linee dall'inizio del programma fino alla linea 50 inclusa.

4. LIST/P 50-

Lista sulla stampante tutte le linee dalla 50 (compresa) fino alla fine del programma.

5. LIST/P

Lista sulla stampante l'intero programma

In modo differito:

6. 10 INPUTX: IF X=1 THEN LIST/P 10 - 100

Lista sulla stampante tutte le linee dalla 10 alla 100 incluse se il valore, digitato in risposta all'istruzione INPUT, è uguale a 1.

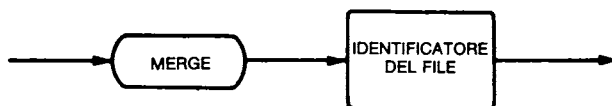
Nota:

— Il numero di linea è una costante numerica che può assumere valori da 0 a 65.535. Il numero di linea 1 deve essere inferiore o uguale al numero di linea 2.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

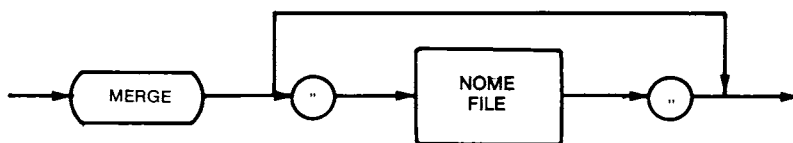
MERGE (Comando)

Esegue l'operazione di MERGE (fusione) tra il programma in memoria e il programma registrato su disco (in formato ASCII).

Formalismo sintattico:

Esempio:

LOAD "A:PROG1"
MERGE "A:PROG2"
SAVE "A:PROG3"

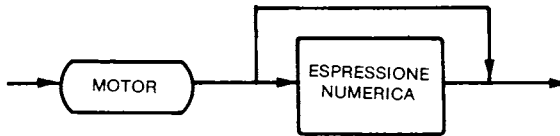
Risultato:
Il comando LOAD carica in memoria il programma PROG1.
Il comando MERGE inserisce il programma PROG2 in PROG1.
Il comando SAVE salva il programma ottenuto dalla fusione con il nome PROG3.

Variante per: MZ-700

Note:
— per IDENTIFICATORE DI FILE vedere appendice A
— il programma su unità esterna deve essere stato salvato in formato ASCII
— la fusione dei due programmi viene effettuata a livello di numero di riga. Se delle righe del file che si sta fondendo hanno lo stesso numero di qualche riga del programma residente in memoria, le righe del file sostituiranno quelle del programma residente in memoria.

MOTOR (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Accende e spegne da programma il motore del registratore a cassetta. Se il valore dell'espressione numerica è zero, il registratore viene spento, se è 1 il registratore viene acceso. Se l'espressione numerica viene omessa, si ottiene un cambiamento di stato del registratore.

Formalismo sintattico:

Esempio:

```
10 MOTOR
20 MOTOR 0
30 MOTOR 1
40 MOTOR 0
```

Risultato:

LINEA 10: poichè il motore era spento, lo accende.
LINEA 20: spegne il motore del registratore.
LINEA 30: riaccende il motore.
LINEA 40: spegne definitivamente il motore del registratore a cassetta.

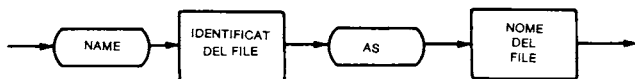
Note:

— ESPRESSIONE NUMERICA può assumere i valori 0 e 1.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

NAME (Comando)

Modifica il nome di un file su disco.

Formalismo sintattico:

Esempio:
NAME "A: PROG.BAS" AS "PIPPO.BAS"

Risultato:
Il file PROG.BAS ha ora il nome PIPPO.BAS.

Note:
— per IDENTIFICATORE DI FILE vedere appendice A.
— dopo l'esecuzione del comando NAME, il nuovo file si ritrova nello stesso minidisco e nella stessa area del vecchio file, ma con il nuovo nome.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

NULL (Istruzione)

Stabilisce il numero di NULL (spazi), dato dal valore dell'espressione numerica considerata, che devono essere stampati alla fine di ogni linea, e ritarda la stampa della riga successiva.

Formalismo sintattico:

Esempio: NULL 3

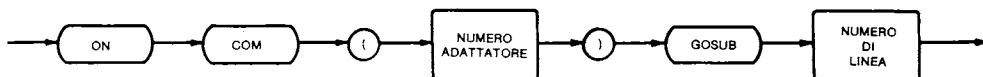
Risultato:
Verranno stampati tre null alla file di ogni riga.

Note:
— il valore di ESPRESSIONE NUMERICA dovrebbe assumere, per un corretto utilizzo delle informazioni, i seguenti valori:
0 solo per nastri non perforati
1 per telescriventi teletype e CRT teletype compatibile
2÷3 per stampanti hard-copy (30 cps)

ON COM (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Definisce un numero di riga del programma a cui passare il controllo dell'esecuzione quando vi sono in arrivo informazioni nella memoria di transito delle comunicazioni.

Formalismo sintattico:

Esempio:

```
 100 COM (1) ON
 110 ON COM (1) GOSUB 1000
     :
 900
1000 REM * INTERCETTAZIONE CARATTERI *
     :
1990 RETURN
```

Risultato:
LINEA 100: attiva l'adattatore per le comunicazioni con numero d'ordine 1.
LINEA 110: sposta il controllo dell'esecuzione alla subroutine che inizia alla linea 100 se vi sono informazioni in arrivo sull'adattatore 1.
LINEE 1000÷1999: sottoprogramma di intercettazione dell'attività di comunicazione per l'adattatore 1.

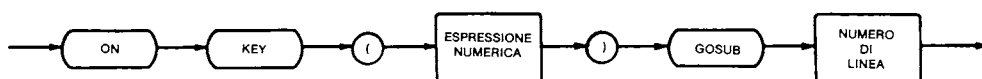
Note:
— NUMERO ADATTATORE è un'espressione numerica che può assumere i valori 1 o 2.
— NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65535.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

ON KEY (Istruzione)

Definisce un numero di riga che deve essere intercettato dal BASIC quando si preme il tasto funzionale specificato dal valore dell'espressione numerica (oppure uno di quelli di controllo del cursore).

Deve essere stata preventivamente eseguita una istruzione KEY ON.

Formalismo sintattico:

Esempio:

```
 100 KEY (7) ON
 110 ON KEY (7) GOSUB 1000
     :
1000 REM * SUBROUTINE PER F7 *
     :
1990 RETURN
```

Risultato:
LINEA 100: attiva l'intercettazione del tasto funzionale 7.

LINEA 110: sposta il controllo dell'esecuzione alla subroutine che inizia alla linea 1000 se è stato premuto il tasto funzione 7

LINEE 1000÷1999: subroutine per il trattamento dovuto all'impostazione del tasto funzione 7.

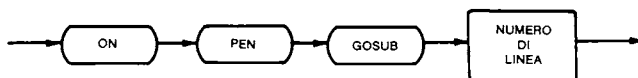
Note:

- NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65.535.
- ESPRESSIONE NUMERICA può assumere i seguenti valori:
  1÷10 tasti funzionali (F1÷F10)
  11 cursore verso l'alto
  12 cursore a sinistra
  13 cursore a destra
  14 cursore verso il basso

ON PEN (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Definisce un numero di riga in modo che il BASIC vi trasferisca il controllo quando si attiva la penna luminosa. Deve essere stata preventivamente eseguita una istruzione PEN ON.

Formalismo sintattico:

Esempio:

```
 100 PEN ON
 110 ON PEN GOSUB 1000
     :
1000 REM * SUBROUTINE PENNA LUMINOSA *
     :
1990 RETURN
```

Risultato:

LINEA 100: attiva la penna luminosa.

LINEA 110: sposta il controllo dell'esecuzione,alla subroutine che inizia alla linea 1000 quando si utilizza la penna luminosa.

LINEE 1000÷1999: subroutine per il trattamento dovuto all'intercettazione della penna luminosa.

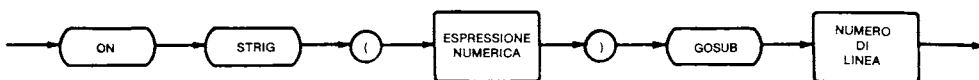
Note:
— NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65.535.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

ON STRIG (Istruzione)

Definisce un numero di riga per l'intercettazione da parte del BASIC quando si preme uno dei tasti sulla barra di comando per i giochi. L'espressione può essere 0, 2, 4, 6 ed indica il tasto da intercettare.

Deve preventivamente essere stata eseguita una STRIG ON.

Formalismo sintattico:

Esempio:

```
 100 STRIG (2) ON
 110 ON STRIG (2) GOSUB 1000
     :
1000 REM * BARRA COMANDO : TASTO B1 *
     :
1990 RETURN
```

Risultato:
LINEA 100: attiva la barra comando giochi (tasto B1)
LINEA 110: sposta il controllo dell'esecuzione alla subroutine che inizia alla linea 1000 quando si preme il tasto B1 della barra comando giochi.
LINEE 1000÷1990: subroutine per il trattamento dovuto alla pressione del tasto B1 della barra comando dei giochi.

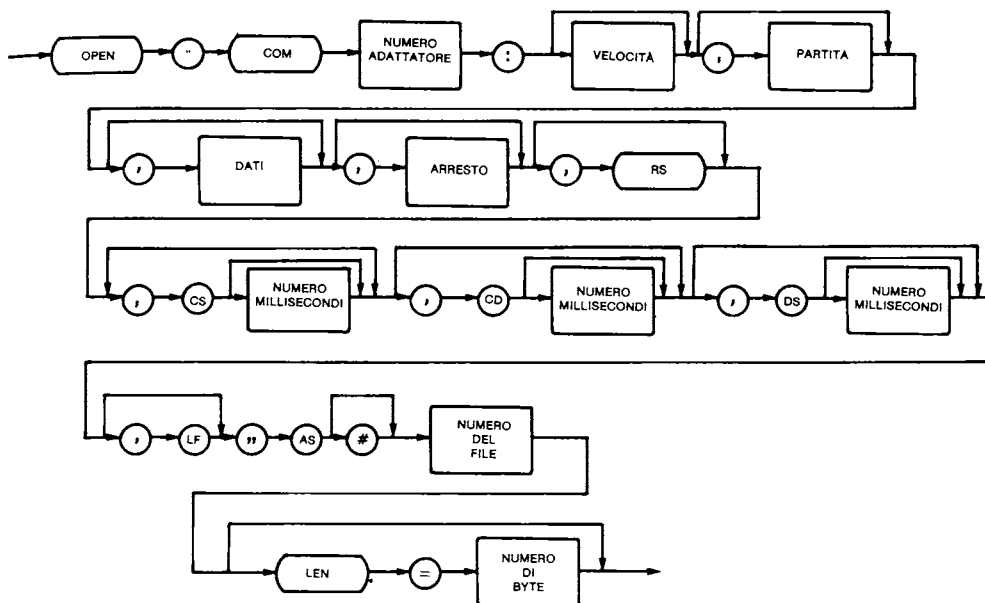
Note:
— NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65.535.
— ESPRESSIONE NUMERICA può assumere i seguenti valori:
0 tasto A1

2 tasto B1
4 tasto A2
6 tasto B2

OPEN "COM (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Apre un file per le comunicazioni.

Formalismo sintattico:

Esempi:

1) OPEN "COM1:" AS #1

2) OPEN "COM2:12000,N,8,,CS,DS,CD" AS # 2

Risultati:

1) Viene aperto il file 1 per le comunicazioni con i valori assunti per difetto. Così la velocità sarà di 300 baud, con controllo di parità, con 7 bit di dati e uno di arresto.
2) Viene aperto il file 2 per le comunicazioni a 12000 baud, senza controllo di parità, con 8 bit di dati e 1 di arresto. Non viene controllato lo stato di CTS, DSR e RLSD.

Note:

— NUMERO ADATTATORE è un'espressione numerica che può assumere i valori 1 e 2.

— VELOCITA' è una costante intera che identifica la velocità di trasmissione in baud (o bps, bit per secondo).

— PARITA' è una costante stringa (1 carattere) che può assumere i seguenti valori:

S SPACE: bit di parità a spazi
O ODD: controllo disparità
M MARK: bit di parità come segno
E EVEN: controllo parità
N NONE: nessun controllo

— DATI è una costante intera che può assumere valori compresi fra 4 e 8. Indica il numero di bit di dati.

— ARRESTO è una costante intera che può assumere i valori 1 e 2. Indica il numero di bit di arresto

— NUMERO MILLISECONDI è un'espressione numerica che può assumere valori compresi fra 0 e 65535

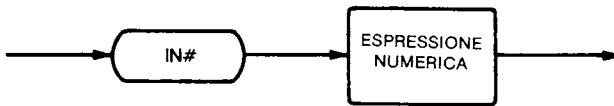
— NUMERO DEL FILE è un'espressione numerica che può assumere valori compresi fra 1 e 3

— NUMERO DI BYTE è un'espressione numerica che identifica il massimo numero di byte della memoria di transito delle comunicazioni che possono essere letti o scritti.

IN# (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Seleziona l'input dalla "slot" specificata dal valore dell'espressione numerica.

Formalismo sintattico:

Esempi:
1) IN# 0
2) IN# 7

Risultati:
1) indica che il successivo ingresso dati avverrà dalla tastiera.
2) indica che il successivo ingresso dati avverrà dalla "slot" numero 7

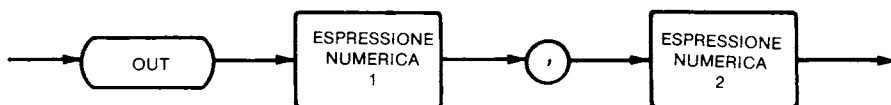
Note:
— Viene utilizzata per definire quale periferica fornirà l'input delle successive istruzioni.
— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 7.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

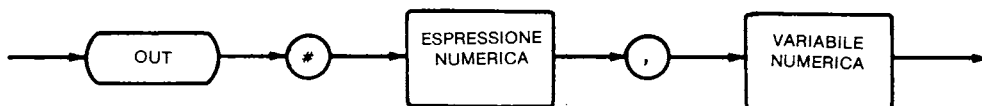
OUT (Istruzione)

Invia un byte all'uscita per l'emissione.

Il valore della prima espressione numerica determina l'uscita di macchina, mentre il valore della seconda espressione numerica rappresenta il dato da trasmettere.

Formalismo sintattico:

Esempi:
1) OUT M , 200
2) OUT M , &HA0

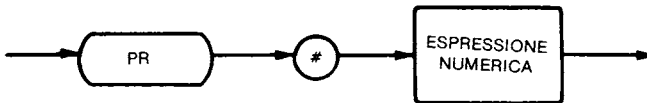
Risultati:
1) invia il byte che contiene il valore decimale 200 all'uscita definita dal contenuto della variabile M
2) invia il byte che contiene il valore esadecimale A0 all'uscita definita dal contenuto della variabile M

Variante per: MZ 700

Note:
— ESPRESSIONE NUMERICA 1 può assumere valori compresi fra 0 e 65535
— ESPRESSIONE NUMERICA 2 può assumere valori compresi tra 0 e 255
— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 255

PR # (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Trasferisce l'output alla "slot" data dal valore dell'espressione numerica.

Formalismo sintattico:

Esempi:
1) PR# 0
2) PR# 7 .

Risultati:
1) riporta l'emissione dei dati verso lo schermo
2) definisce la "slot" 7 come unità di emissione dati

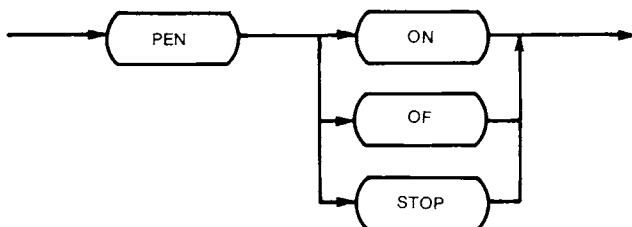
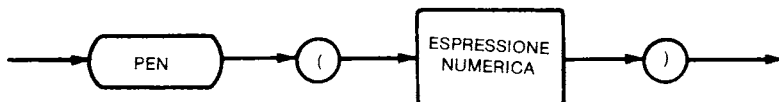
Note:
— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 7
— PR # 0 emette i dati a video.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

PEN (Istruzione e funzione)

Legge la penna luminosa.

Formalismo sintattico:

Come specifica:

Come funzione:

Esempio:

```
  10 PEN ON
  20 ON PEN GOSUB 1000
     :
1000 REM * PENNA LUMINOSA *
1010 A = PEN (0)
1020 B = PEN (3)
1030 PRINT A, B
     :
1990 RETURN
```

Risultato:

LINEA 10: attiva la penna luminosa

LINEA 20: sposta il controllo dell'esecuzione alla subroutine che inizia alla linea 1000 quando viene utilizzata la penna luminosa

LINEA 1010: assegna alla variabile A il valore −1 se la penna è abbassata dall'ultima scansione, altrimenti assegna 0

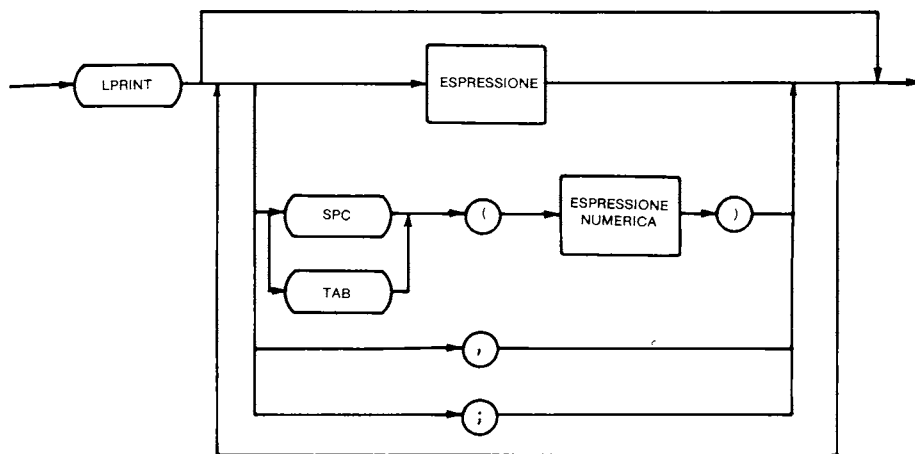
LINEA 1020: assegna alla variabile B il valore −1 se la penna è usata, 0 se non è utilizzata.

Note:
— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 9

LPRINT (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Consente di emettere, su stampante, i dati secondo un formato standard.

Formalismo sintattico:

Esempio:

```
10 A$ = "PIPPO" : B$ = "PLUTO" : C = 10
20 LPRINT A$, B$
30 LPRINT A$; B$
40 LPRINT A$; "E"; B$
50 LPRINT C, A$
```

Risultato:

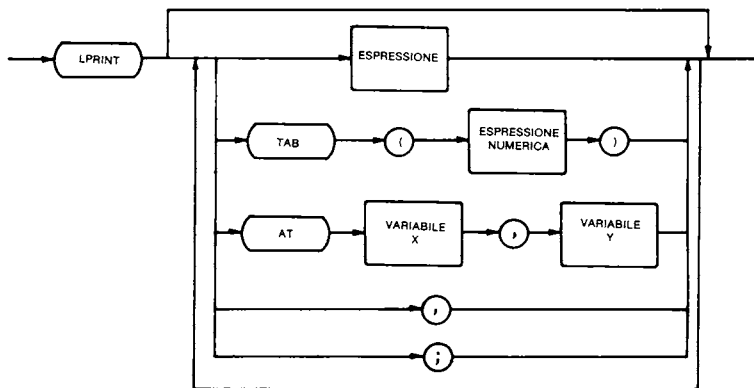
LINEA 10: assegna alle variabili stringa A$ e B$ e alla variabile numerica C i valori specificati.

LINEA 20: stampa sulla stampante le parole:
PIPPO   PLUTO

LINEA 30: stampa sulla stampante le parole:
PIPPOPLUTO

LINEA 40: stampa sulla stampante le parole:
PIPPO E PLUTO

LINEA 50: stampa sulla stampante la configurazione: 10   PIPPO

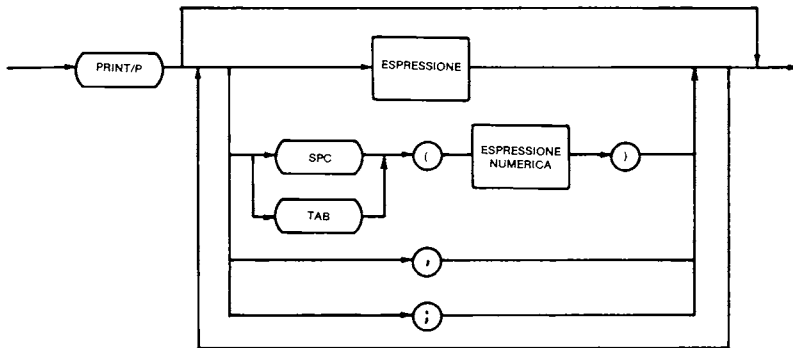
Variante per: ZX81 - ZX Spectrum

Note:

— Se si usa un punto e virgola come separatore, il valore successivo viene stampato immediatamentė dopo il precedente. Se si usa una virgola, il valore successivo viene stampato all'inizio della successiva zona di stampa, che può anche essere alla riga successiva. Se si usa SPC, vengono inseriti tanti spazi quanto è il valore dell'espressione numerica. Se si usa TAB, il successivo valore viene stampato alla colonna determinata dal valore dell'espressione numerica. In entrambi i casi, ciò può far stampare i dati alla riga successiva.

PRINT/P (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Consente di stampare i dati sul plotter secondo un formato standard. (Modo "Testo").

Formalismo sintattico:

Esempio:

```
10 A$ = "PIPPO" : B$ = "PLUTO" : C = 10
20 PRINT/P A$, B$
30 PRINT/P A$; B$
40 PRINT/P A$; "E"; B$
50 PRINT/P C, A$
```

Risultato:

LINEA 10: Assegna alle variabili stringa A$ e B$ rispettivamente i valori PIPPO e PLUTO, e alla variabile numerica C il valore 10

LINEA 20: stampa sulla stampante le parole: PIPPO PLUTO

LINEA 30: stampa sulla stampante le parole: PIPPOPLUTO

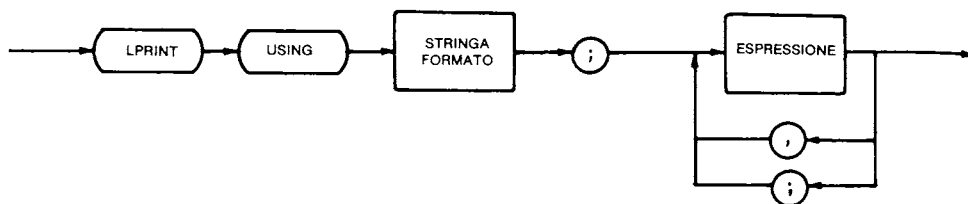
LINEA 40: stampa sulla stampante le parole: PIPPO E PLUTO

LINEA 50: stampa sulla stampante la configurazione: 10 PIPPO

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

LPRINT USING (Istruzione)

Consente di emettere, su stampante, i dati secondo un formato definito dall'utente.

Formalismo sintattico:

Esempio:

```
100 A$ = "TOPO" : B$ = "LINO"
110 LPRINT USING "!"; A$; B$
120 A = 15.7 : B = 9.6 : C = 55.669 : D = - . 157
130 LPRINT USING "# # . # #"; A; B; C; D
140 LPRINT USING "# # . # # -"; A; B; C; D
150 LPRINT USING "+ # # . # #"; A; B; C; D
```

Risultato:

LINEA 100: assegna i valori specificati alle variabili stringa A$ e B$

LINEA 110: stampa il primo carattere del contenuto delle variabili sulla stampante, e cioè TL

LINEA 120: assegna i valori specificati alle variabili numeriche A, B, C, D

LINEA 130: stampa sulla stampante i valori contenuti nelle variabili numeriche nel modo seguente:
15.70 9.60 55.67 0.157

LINEA 140: stampa sulla stampante i valori contenuti nelle variabili numeriche nel modo seguente:
15.70 9.60 55.67 0.157—

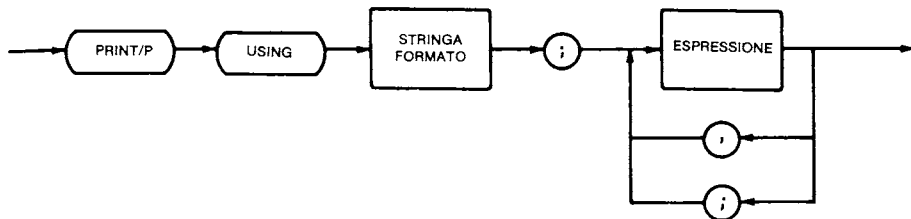
LINEA 150: stampa sulla stampante i valori contenuti nelle variabili numeriche nel modo seguente:
+15.70 +9.60 +55.67 −0.157

Note:
— I caratteri di formattazione utilizzabili all'interno della stringa formato sono parecchi e diversificati nei vari dialetti. Si rimanda alla istruzione PRINT USING e all'appendice A (alla voce STRINGA FORMATO) per la loro spiegazione.

PRINT/P USING (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Consente di emettere, sul plotter, i dati secondo un formato definito dall'utente (Modo "Testo").

Formalismo sintattico:

Esempio:

```
100 A$ = "TOPO" : B$ = "LINO"
110 PRINT/P USING "!"; A$; B$
120 A = 15.7 : B = 9.6 : C = 55.669 : D = −. 157
130 PRINT/P USING "# # . # #"; A; B; C; D
140 PRINT/P USING "# # . # # −"; A; B; C; D
150 PRINT/P USING "+ # # . # #"; A; B; C; D
```

Risultato:

LINEA 100: assegna i valori specificati alle variabili stringa A$ e B$

LINEA 110: stampa il primo carattere del contenuto delle variabili stringa sulla stampante - plotter, e cioè:
TL

LINEA 120: assegna i valori specificati alle variabili numeriche A, B, C, D
LINEA 130: stampa sulla stampante-plotter i valori contenuti nelle variabili numeriche nel modo seguente:
15.70 9.60 55.67 0.157
LINEA 140: stampa sulla stampante-plotter i valori contenuti nelle variabili numeriche nel modo seguente:
15.70 9.60 55.67 0.157 –
LINEA 150: stampa sulla stampante-plotter i valori contenuti nelle variabili numeriche nel modo seguente:
+15.70 +9.60 +55.67 –0.157

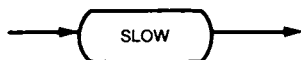
Note:

— I caratteri di formattazione utilizzabili all'interno della stringa formato sono parecchi e diversificati nei vari dialetti. Si rimanda alla istruzione PRINT USING e all'appendice A (alla voce STRINGA FORMATO) per la loro spiegazione.

SLOW (Comando)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Permette di far funzionare lo ZX 81 con la velocità minore. Tale velocità è indicata per fare della grafica.

Formalismo sintattico:

Esempio:

```
100 FAST
110 FOR R = 1 TO 100
120 LET C = 6.28 * R
130 LET A = 3.14 * R ** 2
140 LET V = (3.14 * 4/3) * R ** 3
150 PRINT C, A, V
160 NEXT R
170 SLOW
```

Risultato:

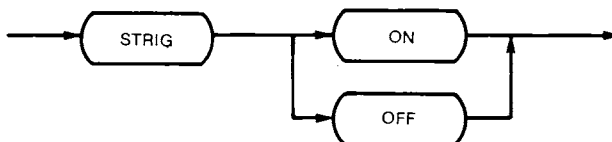
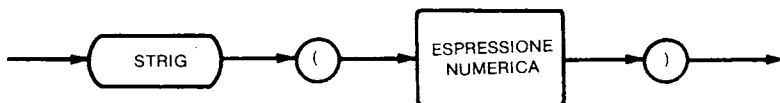
LINEA 100: Imposta la velocità di linea maggiore.
LINEA 110: Inizia un ciclo da ripetersi 100 volte.
LINEE 120÷140: Ad ogni ripetizione calcolano, rispettivamente, il valore della circonferenza, dell'area e del volume della sfera di raggio R (da 1 a 100).
LINEA 150: Stampa i valori calcolati
LINEA 160: Termine del ciclo iniziato alla 110
LINEA 170: Terminato il pezzo di programma contenente parecchi calcoli, ripristina la bassa velocità di esecuzione.

STRIG (Istruzione e funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Ritorna lo stato dei tasti sulle barre di comando per i giochi.

Formalismo sintattico:

Come istruzione:

Come funzione:

Esempio:

```
100 STRIG ON
110 X = STRIG (1)
120 Y = STRIG (3)
130 PRINT X, Y
```

Risultato:

LINEA 100: attiva l'intercettazione della barra comando dei giochi.
LINEA 110: assegna alla variabile X il valore −1 se si sta premendo il tasto A1, altrimenti assegna 0
LINEA 120: assegna alla variabile Y il valore −1 se si sta premendo il tasto B1, altrimenti assegna 0
LINEA 130: stampa il contenuto delle due variabili X e Y.

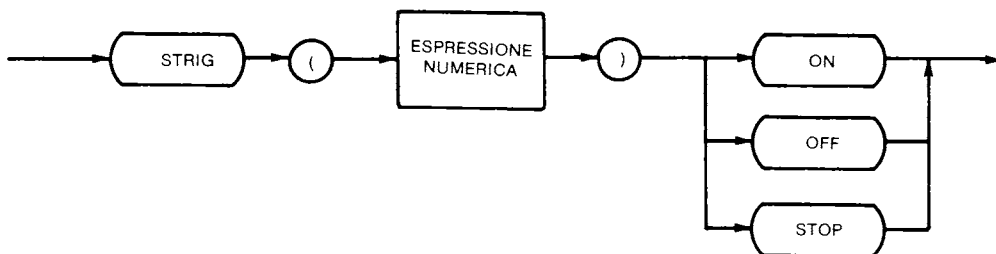
Note:

— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 7.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

STRIG (n) (Istruzione)

Abilita o disabilita l'intercettazione dei tasti sulle barre di comando per i giochi.

Formalismo sintattico:

Esempio:

```
 100 STRIG (2) ON
 110 ON STRIG (2) GOSUB 1000
      :
1000 REM * BARRA COMANDO : TASTO B1 *
      :
1999 RETURN
```

Risultato:

LINEA 100: Attiva la barra comando giochi (Tasto B1).

LINEA 110: Sposta il controllo dell'esecuzione alla subroutine che inizia alla linea 1000 quando si preme il tasto B1 della barra comando giochi.

LINEE 1000÷1999: subroutine per il trattamento dovuto alla pressione del tasto B1 della barra comando dei giochi.

Note:

- NUMERO DI LINEA è un'espressione numerica che può assumere valori compresi fra 0 e 65.535.
- ESPRESSIONE NUMERICA può assumere i seguenti valori:

  0 tasto A1
  2 tasto B1
  4 tasto A2
  6 tasto B2

SWAP (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Permette di scambiare i valori di due variabili semplici.

Le variabili possono essere di qualsiasi tipo, ma devono essere dello stesso tipo. Le variabili devono anche essere già inizializzate.

Formalismo sintattico:

Esempio:

```
100 P$ = "PAOLO" : F$ = "FRANCESCA" : A$ = "AMA"
120 PRINT P$; A$; F$
120 SWAP P$, F$
130 PRINT P$; A$; F$
```

Risultato:

LINEA 100: assegna alle variabili P$, F$ e A$ i valori specificati

LINEA 110: stampa a video la seguente frase:
PAOLO AMA FRANCESCA

LINEA 120: scambia il contenuto delle variabili P$ e F$
LINEA 130: stampa le variabili nello stesso ordine in cui venivano stampate alla linea 110, ma viene ora prodotta la seguente stampa:
FRANCESCA AMA PAOLO

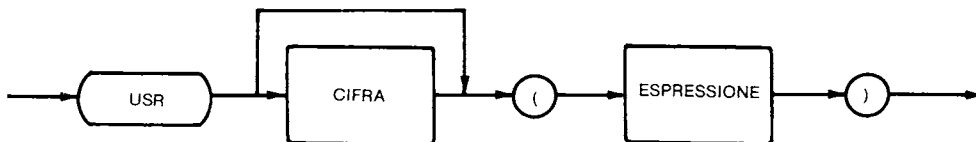
	TI 99/4A
X	VIC 20
X	CBM 64
X	ZX 81
X	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

USR (Funzione)

È utilizzato, nello ZX Spectrum, per memorizzare opportunamente un carattere definito dall'utente. Con argomento numerico è invece utilizzato per eseguire un programma in linguaggio macchina che inizia ad un indirizzo noto.

Per M 20 richiama un sottoprogramma in linguaggio Assembler, passando a questo un argomento. Il valore della cifra corrisponde a quella impostata in una precedente istruzione associata DEF USR.

Per gli altri richiama il programma in linguaggio macchina indicato dal valore dell'espressione numerica.

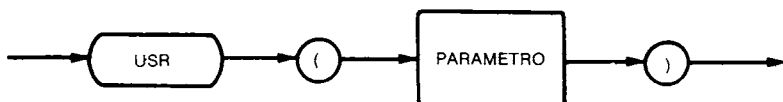
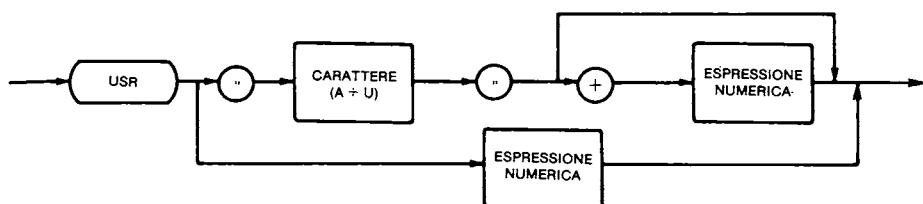
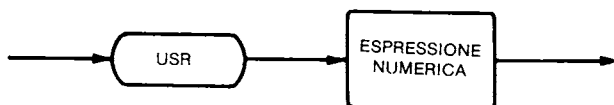
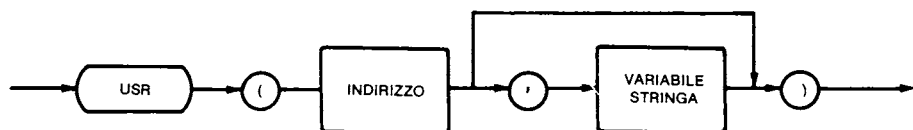
Formalismo sintattico:

Esempio:

```
10 PI = 3.141592
20 INPUT R
30 C = 2 * PI * R
40 DEF SEG 4000
50 DEF USR 0 20000
60 X = USR 0 (C)
```

Risultato:

LINEA 10: assegna alla variabile PI il valore specificato.
LINEA 20: attende che venga digitato un valore numerico a tastiera e lo assegna alla variabile R.
LINEA 30: esegue il calcolo specificato.

LINEA 40: definisce l'attuale segmento di memoria
LINEA 50: definisce l'indirizzo di inizio del sottoprogramma in linguaggio macchina con numero d'ordine 0 come 20.000. L'indirizzo reale di inizio del sottoprogramma 0 sarà all'indirizzo 24.000
LINEA 60: richiama il sottoprogramma 0 in linguaggio macchina, che inizia all'indirizzo 24.000, passando come argomento la variabile C.

Variante per: VIC 20 - CBM 64 - APPLE

Variante per: ZX Spectrum

Variante per: ZX 81

Variante per: MZ 700

Note per IBM P.C.:
— ESPRESSIONE è un'espressione numerica o stringa che sarà l'argomento del sottoprogramma in macchina.

Note per VIC 20 - CBM 64 - APPLE:
— L'indirizzo di partenza del programma scritto in linguaggio macchina è contenuto in locazioni di memoria programmabili (785 e 786 per CBM 64) da impostare con l'istruzione POKE prima dell'utilizzo della funzione USR

Note per ZX Spectrum:
— Viene utilizzata anche per la generazione dei caratteri. Vedere il paragrafo GENERAZIONE CARATTERI e l'istruzione BIN

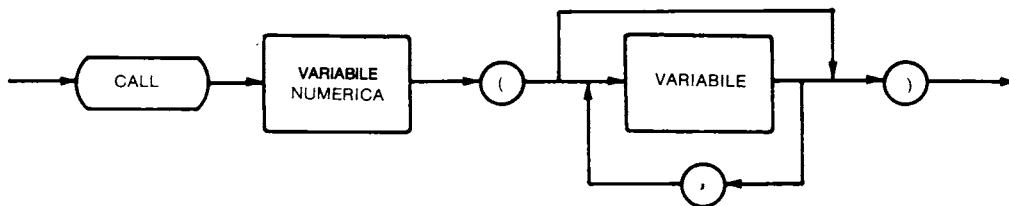
Note per ZX81:
— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 65.535 e rappresenta l'indirizzo di inizio del programma in linguaggio macchina

Note per MZ 700:
— INDIRIZZO è un'espressione numerica che può assumere valori compresi fra 0 e 65.535

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
X	APPLE

CALL (Istruzione)

Per M 20 richiama da BASIC un comando PCOS o un sottoprogramma Assembler, passandogli variabili di programma o costanti (o il valore di una espressione). In I.B.M. P.C. e APPLE II richiama un sottoprogramma in linguaggio macchina.

Formalismo sintattico per: I.B.M. P.C.

Esempio:

```
10 A% = 20000 : B% = 22000
20 PRINT "MENU"
30 PRINT "1 - INSERIMENTO"
40 PRINT "2 - MODIFICA"
50 INPUT "BATTI 1 O 2"; X%
60 IF X% = 1 THEN CALL A%
70 IF X% = 2 THEN CALL B%
```

Risultato:

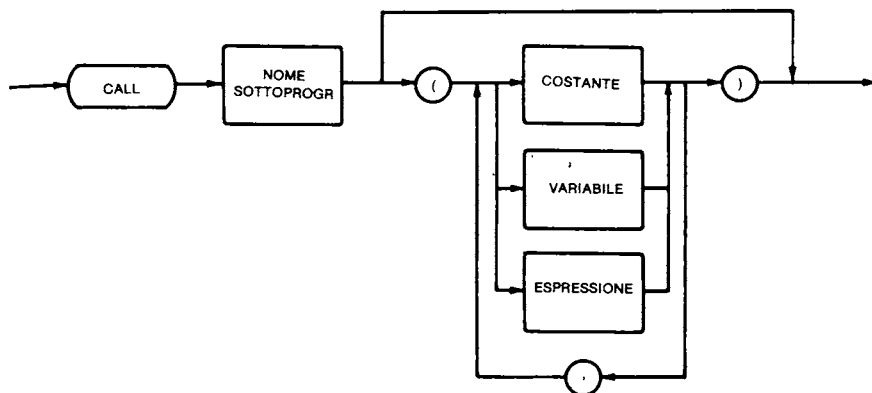
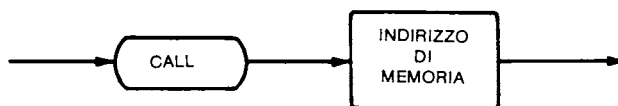
LINEA 10: assegna alle variabili A% e B% i valori specificati

LINEE 20÷40: stampano a video i messaggi specificati

LINEA 50: attende che venga digitato un valore numerico da tastiera, che assegna alla variabile X%

LINEA 60: se il contenuto di X% è 1 richiama il sottoprogramma all'indirizzo 20.000

LINEA 70: se il contenuto di X% è 2 richiama il sottoprogramma all'indirizzo 22.000

Formalismo sintattico per: M 20

Formalismo sintattico per: APPLE

Note:

— per TI99/4A viene utilizzata per richiamare sottoprogrammi di sistema. Tutte le istruzioni vengono elencate altrove separatamente (Esempio CALL CLEAR, CALL SOUND, ecc.). Non esiste invece una istruzione per il richiamo di programmi in linguaggio macchina.

— NOME SOTTOPROGRAMMA è il nome di un comando PCOS o di un sottoprogramma ASSEMBLER; può essere una costante o una variabile stringa.

— INDIRIZZO DI MEMORIA è un'espressione numerica che può assumere valori tra 0 e 65.535

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
X	OLIVETTI M20
	APPLE

EXEC (Istruzione)

Per M 20 richiama da BASIC un comando PCOS o un sottoprogramma Assembler, passando valori costanti; nella espressione stringa comparirà il nome del sottoprogramma seguito da una lista di argomenti costanti.

Formalismo sintattico:

Esempio:

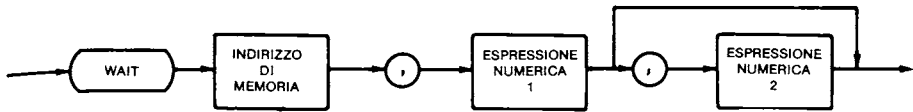
1) EXEC "pk '@', 'INPUT"
2) A$ = "pk '@', 'INPUT": EXEC A$

Risultati:
1) Richiama da BASIC il comando PCOS pkey per assegnare al tasto @ la stringa INPUT. In tal modo è possibile definire dei tasti funzionali.
2) Esegue le stesse funzioni dell'esempio precedente utilizzando la variabile stringa A$.

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
X	APPLE

WAIT (Istruzione)

Sospende l'esecuzione del programma durante il controllo dello stato di un punto di immissione della macchina. L'esecuzione resta sospesa finché il punto di immissione specificato sviluppa una determinata strutura di bit, rappresentata dai valori delle espressioni numeriche.

Formalismo sintattico:

Esempio per CBM 64:
WAIT 53273,6,6

Risultato:
Attende che un'animazione entri in contatto con lo sfondo dello schermo prima di proseguire l'elaborazione.

Esempio per IBM PC:
WAIT 32, 6, 5

Risultato:
Attende fino a che il punto di immissione 32 non abbia una configurazione di bit tale da soddisfare la seguente condizione:
(<CONTENUTO> XOR 5) AND 6 ≠ 0

PAUSE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Sospende l'esecuzione del programma per una durata data dal valore dell'espressione numerica (in cinquantesimi di secondo).

Formalismo sintattico:

Esempio:

```
10 FOR I = 0 TO 7
20 BORDER I
30 PAPER 7 - I
40 INK I
50 CLS : PRINT "PIPPO"
60 PAUSE 50
70 NEXT I
```

Risultato:

LINEA 10: innesca un ciclo di 8 ripetizioni.
LINEA 20: determina il colore del bordo dello schermo.
LINEA 30: assegna il colore allo sfondo dei caratteri.
LINEA 40: assegna il colore al primo piano dei caratteri.
LINEA 50: pulisce il video mostrando i colori impostati e stampa la parola specificata.
LINEA 60: ferma il programma per 1 secondo per permettere la visualizzazione.
LINEA 70: termina il ciclo iniziato alla linea 10.

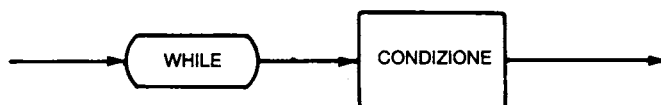
Note:

— ESPRESSIONE NUMERICA può assumere valori compresi tra 0 e 65.535 e individua il numero di scansioni televisive di durata 1/50 di secondo.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

WHILE (Istruzione)

Esegue in ciclo una serie di istruzioni (terminanti con l'istruzione WEND) finché una determinata condizione è vera.

Formalismo sintattico:

Esempio:

```
100 INPUT "DAMMI IL VALORE MASSIMO"; R
110 A = 1
120 WHILE A < R
130 PRINT A
    :
500 A = A + 1
510 WEND
```

Risultato:

LINEA 100: attende che venga digitato un valore numero che assegna alla variabile R

LINEA 110: assegna alla variabile A il valore specificato

LINEA 120: inizio di un ciclo. Il ciclo terminerà quando la condizione diventa falsa

LINEA 130: stampa il valore di A

LINEA 500: aggiunge 1 al contenuto della variabile A.

LINEA 510: termine del ciclo iniziato alla linea 120.

WEND (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Chiude l'elenco delle istruzioni da eseguire nel ciclo iniziato con WHILE.

Formalismo sintattico:

Esempio:

```
100 INPUT "DAMMI IL VALORE MASSIMO"; R
110 A = 1
120 WHILE A < R
130 PRINT A
    :
500 A = A + 1
510 WEND
```

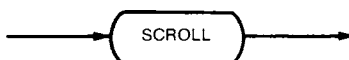
Risultato:

LINEA 100: attende che venga digitato un valore numero che assegna alla variabile R
LINEA 110: assegna alla variabile A il valore specificato
LINEA 120: inizio di un ciclo. Il ciclo terminerà quando la condizione diventa falsa
LINEA 130: stampa il valore di A
LINEA 500: aggiunge 1 al contenuto della variabile A.
LINEA 510: termine del ciclo iniziato alla linea 120.

	TI 99/4A
	VIC 20
	CBM 64
X	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

SCROLL (Comando)

Fa scorrere lo schermo di una linea verso l'alto, perdendo la linea più in alto e liberandone una in basso.

Formalismo sintattico:

Esempio:

```
100 SCROLL
110 PRINT "DIGITA UN NOME"
120 INPUT N$
130 PRINT N$
140 GOTO 100
```

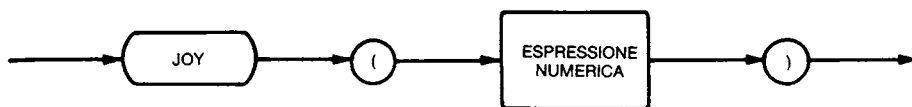
Risultato:

LINEA 100: effettua lo "scroll" del video, fa cioè scorrere lo schermo verso l'alto di una riga
LINEA 110: stampa il messaggio specificato
LINEA 120: attende che venga digitato un valore stringa, che assegna poi alla variabile N$
LINEA 130: stampa il contenuto di N$
LINEA 140: sposta il controllo dell'esecuzione del programma alla linea 100, per iniziare una nuova esecuzione

JOY (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Ritorna un valore che indica la posizione del joystick specificato.

Formalismo sintattico:

Esempio:

```
100 IF (JOY (1) = 0) * (JOY (0) <> 0) THEN T = 0
110 IF (JOY (0) = 255) * (JOY (1) <> 0) THEN T = 1
120 IF (JOY (0) = 0) * (JOY (1) <> 0) THEN T = - 1
```

Risultato:

LINEA 100: valuta la posizione dei due joystick e, se l'insieme delle condizioni è vero, azzera la variabile T

LINEA 110: valuta la posizione dei due joystick e, se l'insieme delle condizioni è vero, assegna il valore 1 alla variabile T.

LINEA 120: valuta la posizione dei due joystick e, se l'insieme delle condizioni è vero, assegna il valore —1 alla variabile T

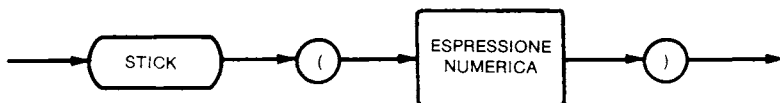
Nota:

— la funzione JOY è presente nella versione 1Z-013B del BASIC dello SHARP MZ-700, ma il manuale in dotazione non ne fà menzione. Sono quindi poche le informazioni di cui siamo in possesso su tale funzione.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

STICK (Funzione)

Ritorna le coordinate X e Y delle due barre di comando per i giochi (joystick).

Formalismo sintattico:

Esempio:

```
100 FOR I = 1 TO 1000
110 X1 = STICK (0)
120 Y1 = STICK (1)
130 X2 = STICK (2)
140 Y2 = STICK (3)
150 CLS
160 LINE (X1, Y1) – (X2, Y2)
170 NEXT I
```

Risultato:

LINEA 100: inizia un ciclo da ripetere 1000 volte

LINEE 110÷140: le coordinate X e Y dei due joystick vengono assegnate alle variabili specificate

LINEA 150: pulisce il video

LINEA 160: traccia una linea tra il punto di coordinate (X1, Y1), che dipende quindi dalla posizione del Joystick A, al punto di coordinate (X2, Y2), che dipende quindi dal joystick B. La linea si modificherà continuamente manipolando i due joystick.

LINEA 170: termina il ciclo iniziato alla linea 100

Note:

— ESPRESSIONE NUMERICA può assumere i seguenti valori:

0 ritorna la coordinata X del joystick A
1 ritorna la coordinata Y del joystick A
2 ritorna la coordinata X del joystick B
3 ritorna la coordinata Y del joystick B

PDL (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
X	APPLE

Dà il valore corrente del controllo dei sensori (o PADDLE) specificato dal valore dell'espressione numerica.

Formalismo sintattico:

Esempio:

```
  90 REM * VELOCITA' DELL'ASTRONAVE *
 100 SPEED = PDL (3)
 110 GO TO 1000
      :
1000 REM * ALTA VELOCITA' *
```

Risultato:

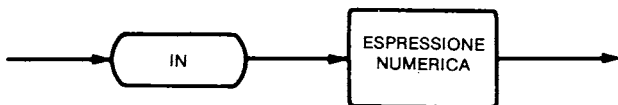
LA LINEA 90 documenta il programma
LA LINEA 100 assegna la velocità alla quale i caratteri devono essere inviati al video: essa dipende dal valore intercettato sul PADDLE 3.
LA LINEA 110 salta alla 1000

Note:
— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 3
— il PADDLE è una resistenza variabile tra 0 e 150 KΩ

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
X	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

IN (Funzione)

Ritorna il byte letto dall'uscita definita dal valore dell'espressione numerica.

Formalismo sintattico:

Esempio:

```
10 FOR I = 1 TO 10
20 A = IN 50 + 1
30 PRINT A
40 NEXT I
```

Risultato:

LINEA 10: inizia un ciclo che verrà ripetuto 10 volte

LINEA 20: assegna alla variabile A il valore letto dalla porta specificata dal valore dell'espressione numerica (cioè da 51 a 60, in base alle ripetizioni).

LINEA 30: stampa il valore contenuto nella variabile A.

LINEA 40: termine del ciclo iniziato alla linea 10.

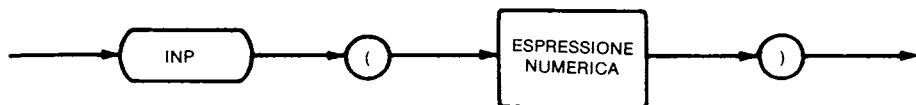
Note:

— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 65.535, ed identifica il valore della porta di I/O dalla quale leggere il byte

INP (Funzione - istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Ritorna il byte letto dall'uscita definita dal valore dell'espressione numerica. In MZ 700 il valore letto viene assegnato alla variabile numerica.

Formalismo sintattico per: I.B.M. P.C. (funzione)

Esempio:

```
100 FOR I = 1 TO 10
110 A = INP (250 + I)
120 PRINT A
130 NEXT I
```

Risultato:

LINEA 100: inizia un ciclo che verrà ripetuto 10 volte

LINEA 110: assegna alla variabile A il valore letto dalla porta specificata dal valore dell'espressione numerica (cioè da 251 a 260, in base alle ripetizioni).

LINEA 120: stampa il valore contenuto nella variabile A

LINEA 130: termine del ciclo iniziato alla linea 10

Formalismo sintattico per: MZ 700 (istruzione)

Note:

— ESPRESSIONE NUMERICA può assumere valori compresi fra 0 e 65.535 per IBM PC e fra 0 e 255 per SHARP MZ-700

	TI 99/4A
X	VIC 20
X	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

SYS (Funzione - istruzione)

È una funzione che trasferisce il controllo del programma a un sottoprogramma indipendente.

Formalismo sintattico:

Esempio:

10 SYS 28990

Risultato:

L'istruzione SYS trasferisce il controllo del programma all'istruzione in linguaggio macchina presente all'indirizzo 28990

Note:

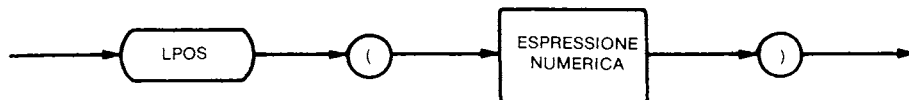
— INDIRIZZO DI MEMORIA è un'espressione numerica che può assumere valori compresi fra 0 e 65535

— Il programma in linguaggio macchina richiamato con una SYS deve terminare con la istruzione RTS (Return), in modo che l'elaborazione, alla fine del programma in linguaggio macchina, ritorni al programma BASIC, all'istruzione successiva alla SYS.

LPOS (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

Ritorna la posizione attuale della testina di stampa nella memoria di transito della stampatrice.

Formalismo sintattico:

Esempio:

```
 100 GOSUB 1000
 110 LPRINT A$; A, B
 120 IF LPOS (1) > 80 GOSUB 2000
 130 GOTO 100
      :
1000 REM * ASSEGNAZIONE VALORI *
1010 INPUT "NOME"; A$
1020 INPUT "ETA'"; A
1020 INPUT "STIPENDIO"; B
1030 RETURN
      :
2000 REM * FINE RIGA *
2010 LPRINT CHR$ (13)
2020 RETURN
```

Risultato:

LINEA 100: manda in esecuzione la subroutine che inizia alla linea 1000

LINEA 110: stampa il contenuto delle variabili specificate sulla stampante collegata (LPT1:).

LINEA 120: se la lunghezza della riga supera gli 80 caratteri, manda in esecuzione la subroutine che inizia alla linea 2000.

LINEA 130: sposta il controllo dell'esecuzione alla linea 100, per eseguire di nuovo il programma

LINEE 1000÷1030: subroutine per l'acquisizione dei valori da stampare.

LINEE 2000÷2020: permettono di stampare, a fine riga, un carattere di ritorno a margine in modo da saltare alla riga successiva.

Note:

— ESPRESSIONE NUMERICA indica la stampante interessata, e può assumere i seguenti valori:

0 o 1	LPT1:
2	LPT2:
3	LPT3:

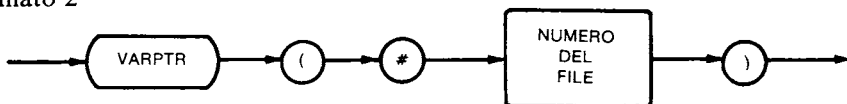
	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
X	OLIVETTI M20
	APPLE

VARPTR (Funzione)

Nel formato 1 fornisce l'indirizzo di memoria del primo byte del dato associato alla variabile specificata. Nel formato 2 fornisce l'indirizzo iniziale del buffer di I/O associato al file che, per file random, è definito tramite l'istruzione FIELD.

Formalismo sintattico:

Formato 1

Formato 2

Esempio:

```
10 A = 1000
20 I = VARPTR (A)
30 OPEN "PIPPO" AS # 1
40 GET # 1
50 F = VARPTR (# 1)
```

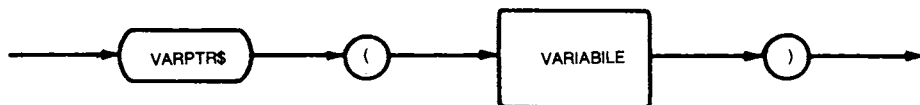
Risultato:

LINEA 10: assegna alla variabile A il valore specificato
LINEA 20: assegna alla variabile I l'indirizzo di memoria del primo dei 4 byte che compongono la variabile (a semplice precisione) I
LINEA 30: apre il file "PIPPO" col numero d'ordine 1
LINEA 40: scrive un record sul file 1
LINEA 50: assegna alla variabile F l'indirizzo iniziale della memoria di transito associata al file 1.

VARPTR$ (Funzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
	MZ-700
X	IBM P.C.
	OLIVETTI M20
	APPLE

Ritorna l'indirizzo di una variabile in memoria, sottoforma di un carattere. Viene utilizzata in I.B.M. P.C. principalmente con PLAY e DRAW nei programmi che saranno compilati successivamente.

Formalismo sintattico:

Esempio:

```
100 SCALA = 10
110 A$ = "E15 F15 L30"
120 DRAW "X" + VARPTR$ (A$)
130 DRAW "S =" + VARPTR$ (SCALA)
```

Risultato:

LINEA 100: assegna alla variabile SCALA il valore specificato.
LINEA 110: assegna alla stringa A$ il valore specificato.
LINEA 120: disegna il contenuto della stringa A$. È equivalente alla: DRAW "X A$".
LINEA 130: assume il nuovo valore di scala contenuto nella variabile SCALA. È equivalente alla: DRAW "S = SCALA".

CAPITOLO 15

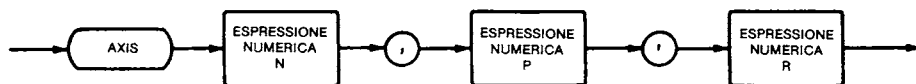
ISTRUZIONI PER LA STAMPANTE - PLOTTER DELLO SHARP MZ - 731

AXIS
CIRCLE
GPRINT
HSET
LINE
RLINE
MODE
MOVE
RMOVE
PAGE
PCOLOR
PHOME
PLOT
SKIP
TEST

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

AXIS (Istruzione)

Traccia sul plotter l'asse delle Y (quando il valore dell'espressione numerica N è = 0) o l'asse delle X (quando il valore dell'espressione numerica N è = 1). I numeri di segni di scala sono dati dal valore dell'espressione numerica R e vengono tracciati con un passo dato dal valore dell'espressione numerica P. (Modo "Grafico").

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 MOVE 240, 0
30 AXIS 0, - 10, 48
40 MOVE 0, - 240
50 AXIS 1, 10, 48
60 MODE TN
```

Risultato: vengono tracciati gli assi cartesiani con i segni di scala da −240 a 240 ad intervalli di 10 unità.

LINEA 10: commuta la stampante al modo operativo grafico.
LINEA 20: Alza la penna e la sposta alla posizione (240,0).
LINEA 30: Traccia l'asse delle Y includendo i segni di scala ad intervalli di 10 unità.
LINEA 40: Alza la penna e la sposta alla posizione (0, −240).
LINEA 50: traccia l'asse delle X includendo i segni di scala ad intervalli di 10 unità.
LINEA 60: Riporta la stampante al modo operativo testo.

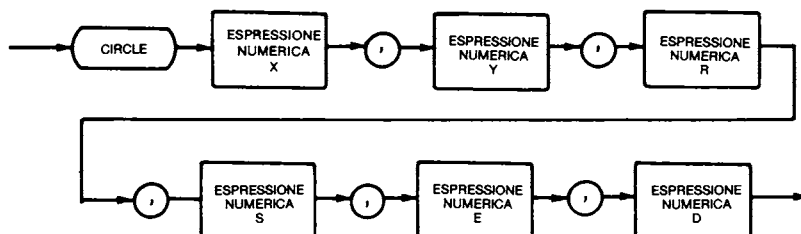
Note:

— il valore dell'espressione numerica P (passo di scala) può variare da −999 a 999
— il valore dell'espressione numerica R (numero di ripetizioni) può variare da 1 a 255.

CIRCLE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Traccia un cerchio o un arco con un raggio dato dal valore dell'espressione numerica R con passo dato dall'espressione numerica D e centro dato dalle coordinate X e Y, partendo dall'angolo dato dal valore dell'espressione numerica S e terminando all'angolo dato dal valore dell'espressione numerica E. Viene tracciato un cerchio completo quando:
S = 0 ; E = 360 ; D = 0.2
(Modo "Grafico")

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 MODE 240, - 240
30 HSET
40 CIRCLE 0, 0, 240, 0, 360, 0.2
50 CIRCLE 240, 0, 240, 90, 270, 0.2
60 MODE TN
```

Risultato:

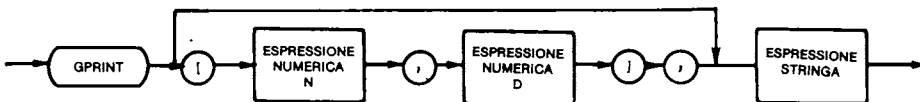
LINEA 10: commuta la stampante al modo operativo grafico.
LINEA 20: Alza la penna e la sposta alla posizione (240, −240) (al centro della carta).
LINEA 30: Fissa la posizione attuale (al centro della carta) come nuova origine delle coordinate.
LINEA 40: Traccia un cerchio di raggio 240 unità, con centro nella nuova origine e con passo 0.2.
LINEA 50: Traccia un semicerchio di raggio 240 unità, con centro in posizione (240,0) rispetto alla nuova origine, iniziando da 90° e terminando a 270°, con passo 0.2.
LINEA 60: Riporta la stampante al modo operativo testo.

Note:
— Il valore dell'espressione numerica X e Y (coordinate del centro) può variare da −999 a 999
— Il valore dell'espressione numerica R (raggio) può variare da zero a 999
— Il valore delle espressioni numeriche S (angolo iniziale), E (angolo finale) e D (angolo di traslazione) può variare da 0 a 999.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

GPRINT (Istruzione)

Stampa su plotter la stringa di caratteri specificata con dimensione e direzione voluti. Il valore dell'espressione numerica N indica la dimensione del carattere, mentre il valore dell'espressione numerica D indica la direzione del carattere. ("Modo Grafico").

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 GPRINT "A"
30 GPRINT [ 2,2 ], "A"
40 MODE TN
```

Risultato:
LINEA 10: Commuta la stampante-plotter al modo operativo grafico.
LINEA 20: Stampa il carattere "A" nel modo operativo grafico.
LINEA 30: Stampa una A capovolta (D=2) nel modo 26 caratteri per riga (N=2).
LINEA 40: Riporta la stampante-plotter al modo operativo testo.

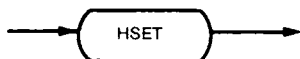
Note:
— L'espressione numerica N (dimensione del carattere) può assumere i valori da 0 a 63, e precisamente:
0 = equivalente a 80 caratteri per riga
1 = equivalente a 40 caratteri per riga
2 = equivalente a 26 caratteri per riga
>2 = meno di 26 caratteri per riga

— l'espressione numerica D (direzione del carattere) può assumere i valori da 0 a 3, e precisamente:
0 = direzione normale
1 = carattere accosciato a destra
2 = carattere capovolto
3 = carattere accosciato a sinistra

HSET (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Stabilisce che la posizione attuale della penna del plotter diventi la nuova origine. In tal modo è possibile fissare l'origine nella posizione più appropriata per disegnare figure ("Modo Grafico").

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 MOVE 240, - 500
30 HSET
40 CIRCLE 0, 240, 240, 180, 360, 0.2
50 MODE TN
```

Risultato:

LINEA 10: Commuta la stampante-plotter al modo operativo grafico.

LINEA 20: Alza la penna e la sposta alla posizione (240, −500), cioè in centro al foglio e, in basso, per metà escursione dell'asse Y.

LINEA 30: Fissa la posizione attuale della penna, e cioè (240, −500) rispetto alla posizione iniziale, come nuova origine. Tutti gli spostamenti successivi saranno quindi relativi a questa posizione, che diventa quindi la posizione (0,0)

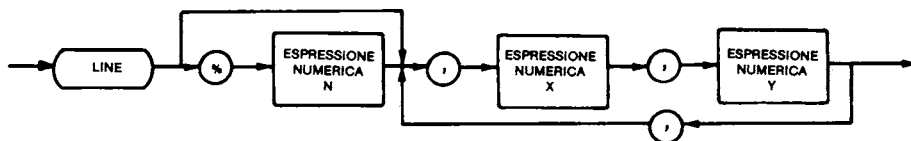
LINEA 40: traccia un semicerchio con raggio 240 unità in posizione (0,240) rispetto alla nuova origine; tale semicerchio passerà dall'origine stessa.

LINEA 50: Riporta la stampante-plotter al modo operativo testo.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

LINE (Istruzione)

Questa istruzione traccia una riga, sul plotter, dalla posizione attuale della penna fino al punto con coordinate assolute X e Y, poi, successivamente, da questo ad un'altro punto con coordinate assolute diverse, e così via. La linea può essere intera o tratteggiata in funzione del valore assunto dall'espressione numerica N. (1 = intera, 2 ÷ 16 = tratteggiata) ("Modo Grafico").

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 LINE 480, 0, 480, - 480, 0, - 480, 0, 0
30 MOVE 0, - 500
40 HSET
50 H = INT (120 * SQR (3))
60 LINE %2, 240, 0, 120, — H, 0, 0
70 MODE TN
```

Risultato:

LINEA 10: Commuta la stampante-plotter al modo operativo grafico.

LINEA 20: Traccia una linea intera dall'origine (0,0) alla posizione (480,0), poi da questa alla posizione (480, −480), da questa ancora alla posizione (0, −480) e successivamente traccia un'ultima riga fino all'origine, cioè alla posizione (0,0). Il risultato è un quadrato di 480 unità per lato.

LINEA 30: Alza la penna e la sposta alla posizione (0, −500).

LINEA 40: Fissa la nuova origine.

LINEA 50: Calcola il valore di H (altezza).

LINEA 60: Traccia un triangolo equilatero rovesciato utilizzando una linea fine-

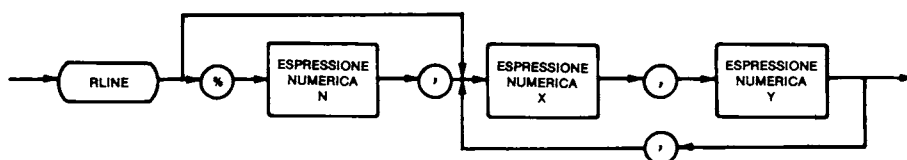
mente tratteggiata (%2).
LINEA 70: Riporta la stampante al modo operativo testo.

Note:
— L'espressione numerica X può assumere valori da −480 a 480
— L'espressione numerica Y può assumere valori da −999 a 999

RLINE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Questa istruzione traccia una riga, sul plotter, dalla posizione attuale della penna fino al punto con coordinate relative X e Y, poi, successivamente, da questo ad un altro punto con nuove coordinate relative, e così via.
La linea può essere intera o tratteggiata in funzione del valore assunto dall'espressione numerica N (1 = intera, 2 ÷ 16 = tratteggiata) (Modo "Grafico").

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 RLINE 480, 0, 0, - 480, - 480, 0, 0, 480
30 MOVE 0, - 500
40 HSET
50 H = INT (120 * SQR (3))
60 RLINE %2, 240, 0, - 120, - H, - 120, H
70 MODE TN
```

Risultato: è lo stesso dell'esempio presente all'istruzione LINE, e cioè:

LINEA 10:	Commuta la stampante-plotter al modo operativo grafico.
LINEA 20:	Traccia un quadrato, con linea intera, di 480 unità per lato.
LINEA 30:	Alza la penna e la sposta alla posizione (0, −500).
LINEA 40:	Fissa la nuova origine.
LINEA 50:	Calcola il valore di H (altezza)
LINEA 60:	Traccia un triangolo equilatero rovesciato, di altezza H, utilizzando una linea con tratteggio molto fine (%2).
LINEA 70:	Riporta la stampante al modo operativo testo.

Note:

— L'espressione numerica X può assumere valori da −480 a 480

— L'espressione numerica Y può assumere valori da −999 a 999.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

MODE (Comando)

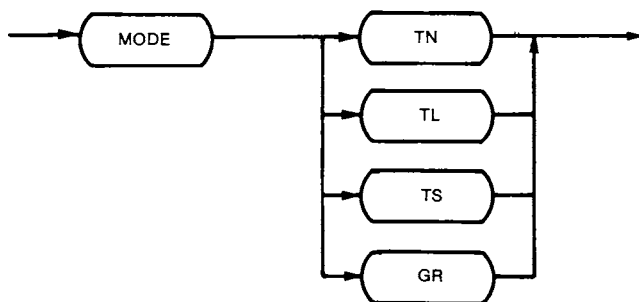
Viene usato per cambiare il modo operativo della stampante - plotter a colori.

TN = modo operativo testo, 40 caratteri per riga

TL = modo operativo testo, 26 caratteri per riga

TS = modo operativo testo, 80 caratteri per riga

GR = modo operativo grafico

Formalismo sintattico:

Esempio:

```
10 PRINT/P "MODO STANDARD"
20 MODO TL
30 PRINT/P "MODO 26 CARATTERI PER RIGA"
40 MODE TS
50 PRINT/P "MODO 80 CARATTERI PER RIGA"
60 MODE GR
70 GPRINT "MODO GRAFICO"
80 MODE TN
```

Risultato:

LINEA 10: Stampa sulla stampante la stringa seguente utilizzando 40 caratteri per riga (modo testo standard)

LINEA 20: Commuta la stampante-plotter al modo testo che utilizza 26 caratteri per riga.

LINEA 30: Stampa sulla stampante la stringa evidenziata utilizzando 26 caratteri per riga.

LINEA 40: Commuta la stampante-plotter al modo testo che utilizza 80 caratteri per riga.

LINEA 50: Stampa sulla stampante la stringa evidenziata, utilizzando 80 caratteri per riga.

LINEA 60: Commuta la stampante-plotter al modo grafico.

LINEA 70: Stampa sulla stampante la stringa evidenziata, in modo grafico (con 40 caratteri per riga).

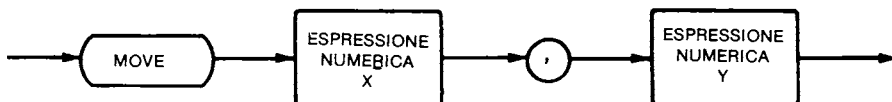
LINEA 80: Riporta la stampante plotter al modo operativo testo standard (40 caratteri per riga).

Note:

— All'atto dell'accensione viene impostato il modo operativo testo con 40 caratteri per riga.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

MOVE (Istruzione)

Fa alzare la penna e la sposta alla posizione specificata dalle coordinate assolute X e Y (Modo "Grafico")

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 MOVE 0, - 500
30 HSET
40 LINE 480, 0
50 MOVE 240, 240
60 LINE 240, - 240
70 MODE TN
```

Risultato: disegna una croce al centro del foglio.

LINEA 10: Commuta la stampante-plotter al modo operativo grafico.
LINEA 20: Alza la penna e la sposta alla posizione (0, −500)
LINEA 30: Fissa la posizione attuale come nuova origine
LINEA 40: Traccia una linea dall'origine alla posizione (480,0)
LINEA 50: Alza la penna e la sposta alla posizione (240, 240)
LINEA 60: Traccia una linea dalla posizione attuale (240, 240) alla posizione (240, −240)
LINEA 70: Riporta la stampante-plotter al modo operativo testo

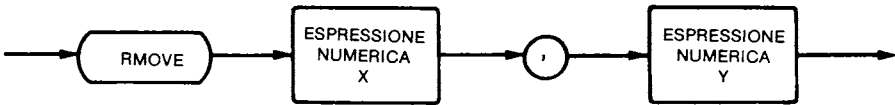
Note:

— L'espressione numerica X può assumere valori da —480 a 480.
— L'espressione numerica Y può assumere valori da −999 a 999.

RMOVE (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Fa alzare la penna e la sposta alla posizione specificata dalle coordinate relative X e Y. (Modo "Grafico")

Formalismo sintattico:

Esempio:

```
10 MODE GR
20 MOVE 0, – 500
30 HSET
40 LINE 480, 0
50 RMOVE – 240, 240
60 LINE 240, – 240
70 MODE TN
```

Risultato: disegna la stessa croce vista nell'esempio dell'istruzione MOVE

LINEA 10: Commuta la stampante-plotter al modo operativo grafico.
LINEA 20: Alza la penna e la sposta alla posizione (0, –500)
LINEA 30: Fissa la posizione attuale come nuova origine
LINEA 40: Traccia una linea dall'origine alla posizione (480,0)
LINEA 50: Alza la penna e la sposta di –240 unità nella direzione X e di 240 unità nella direzione Y
LINEA 60: Traccia una linea dalla posizione attuale (240, 240) fino alla posizione (240, –240)
LINEA 70: Riporta la stampante-plotter al modo operativo testo

Note:
— L'espressione numerica X può assumere valori da –480 e 480
— L'espressione numerica Y può assumere valori da –999 a 999.

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

PAGE (Comando)

Specifica il numero di righe per pagina. (Modo "Testo").

Formalismo sintattico:

Esempio: in modo immediato:

1. PAGE 72

Considera la pagina composta di 72 righe.

2. PAGE 50

Considera la pagina composta di 50 righe.

3. PAGE 1

Considera la pagina composta di 1 riga, per cui effettua il salto a pagina nuova dopo ogni riga stampata

In modo differito:

```
10 INPUT "QUANTE RIGHE PER PAGINA?" ; R
20 IF (R < 1) + (R > 72) THEN END
30 PAGE R
40 PRINT/P R
```

Risultato:

LINEA 10: Richiede all'utente il numero di righe per pagina volute e memorizza la risposta nella variabile R.

LINEA 20: Se il valore di R non è compreso nei limiti, il programma viene fatto terminare.

LINEA 30: Imposta il numero di righe per pagina in base al contenuto di R.

LINEA 40 e seguenti: Stampa informazioni sulla stampante e proseguimento del programma.

Note:

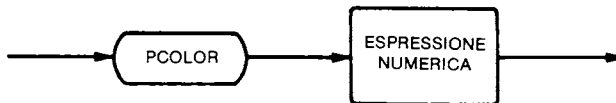
— All'atto dell'accensione viene impostato il valore di 60 righe per pagina. Ogni riga ha una altezza di circa 4/21 di pollice, contrariamente all'altezza standard delle righe di stampa che è di 1/6 di pollice.

— Il valore dell'espressione numerica può essere compreso tra 1 e 72

PCOLOR (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Specifica il colore da usare per la stampa dei caratteri o dei grafici. Il valore dell'espressione numerica deve essere compreso tra 0 e 3 (0 = nero, 1 = blu, 2 = verde, 3 = rosso).

Formalismo sintattico:

Esempi: in modo immediato:
1. PCOLOR 3
Indica che le prossime informazioni verranno stampate in rosso.
2. PCOLOR 1
Indica che le successive informazioni verranno stampate in blu

In modo differito:

```
10 A = 5 : B = 10 : C = 1
20 MODE GR
30 PCOLOR (B/A) + C
40 CIRCLE 0, 0, 240, 0, 360, 0.2
50 MODE TN
60 PCOLOR 0
```

Risultato:

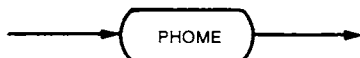
LINEA 10: assegnazione valori alle variabili A, B, C
LINEA 20: Commuta la stampante al modo operativo grafico.
LINEA 30: Determina il colore da utilizzare per le stampe successive. Il risultato dell'espressione numerica è uguale a 3 (rosso)
LINEA 40: Traccia un cerchio di colore rosso
LINEA 50: Riporta la stampante al modo operativo testo
LINEA 60: Riporta il plotter al colore base (nero)

Note:
— all'atto dell'accensione viene impostato il colore nero
— il comando può essere usato sia in modo grafico che in modo testo
— il valore dell'espressione numerica deve essere compreso tra 0 e 3

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

PHOME (Istruzione)

Riporta la penna del plotter all'origine (coordinate assolute 0,0) (Modo "Grafico").

Formalismo sintattico:

Esempio:

```
 10 MODE GR
 20 MOVE 0, - 500
 30 HSET
 40 LINE 480, 0
 50 MOVE 240, 240
 60 LINE 240, - 240
 70 PHOME
 80 PCOLOR 2
 90 LINE 0, 240, 480, 240, 480, - 240, 0, - 240, 0, 0
100 MODE TN
```

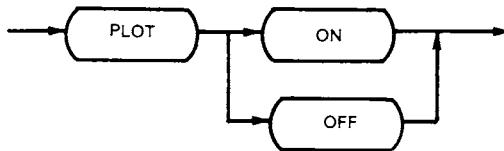
Risultato: disegna una croce in nero al centro del foglio, inserita in un quadrato verde.

LINEA 10: Commuta la stampante-plotter al modo operativo grafico.
LINEA 20: Alza la penna e la sposta alla posizione (0, −500)
LINEA 30: Fissa la posizione attuale come nuova origine
LINEA 40: Traccia una linea orizzontale dall'origine alla posizione (480,0)
LINEA 50: Alza la penna e la sposta alla posizione (240, 240)
LINEA 60: Traccia una linea verticale dalla posizione attuale (240, 240) alla posizione (240, −240)
LINEA 70: Riporta la penna all'origine
LINEA 80: Imposta il colore verde
LINEA 90: Partendo dall'origine traccia un quadrato (in verde) di 480 unità di lato, e che comprende la croce disegnata in precedenza.
LINEA 100: Riporta la stampante-plotter al modo operativo testo.

PLOT (Istruzione)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Rende possibile l'uso della stampante plotter come unità video. (Modo "Testo").

Formalismo sintattico:

Esempio:

```
10 MODE TN
20 PLOT ON
30 PRINT "USO IL PLOTTER AL POSTO DEL VIDEO"
40 CURSOR 5, 1
50 PRINT "COGNOME"
60 CURSOR 5, 10
70 INPUT C$
80 PLOT OFF
```

Risultato:

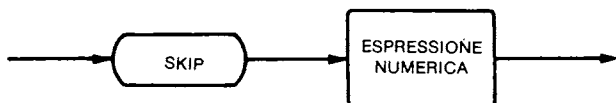
LINEA 10: Commuta la stampante-plotter al modo operativo testo.
LINEA 20: Abilita la stampante a funzionare come unità video
LINEA 30: Stampa sul plotter la frase specificata
LINEA 40: Sposta il cursore alla posizione specificata (sulla carta del plotter)
LINEA 50: Stampa sul plotter la frase specificata
LINEA 60: Sposta il cursore alla posizione specificata (sulla carta del plotter)
LINEA 70: Aspetta che l'utente introduca dei dati alfanumerici (il cognome)
LINEA 80: Disabilita la stampante a funzionare come unità video

Note:
- Eventuali caratteri non previsti nel set della stampante verranno sostituiti da punti
- Deve sempre venir precedentemente eseguita una istruzione MODE TN, prima di poter utilizzare l'istruzione PLOT ON
- Si può usare CTRL + G per effettuare il cambio della penna
- I tasti INST, DEL e ← vengono disabilitati quando si esegue una istruzione PLOT ON

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

SKIP (Comando)

Viene usato per trascinare la carta del plotter.
Il trascinamento avviene nella direzione di avanzamento e di N righe (dove N è il valore dell'espressione numerica) quando N è positivo, nella direzione opposta quando N è negativo. (Modo "Testo").

Formalismo sintattico:

Esempi: in modo immediato:

1. SKIP 10

La carta viene trascinata di 10 righe in avanti

2. SKIP −10

La carta viene trascinata di 10 righe nella direzione contraria a quella di avanzamento

In modo differito:

```
10 MODE TL
20 PRINT/P "INTESTAZIONE"
30 INPUT "QUANTE RIGHE DI SPAZI?"; R
40 IF (R < − 20) + (R > 20) THEN END
50 SKIP R
60 PRINT/P "ELENCO DATI"
```

Risultato:

LINEA 10: commuta la stampante-plotter al modo operativo testo, con 26 caratteri per riga
LINEA 20: stampa sul plotter la frase considerata
LINEA 30: richiede all'utente il numero di righe di cui deve essere trascinata la carta, e memorizza la risposta nella variabile R
LINEA 40: controlla che R sia compresa nei limiti
LINEA 50: esegue il trascinamento della carta per R righe
LINEA 60: stampa sul plotter la frase considerata

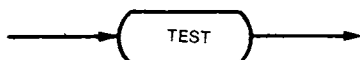
Note:
— il valore dell'espressione numerica deve essere compreso fra —20 e 20

— quando si utilizzano valori negativi dell'espressione numerica, bisogna accertarsi che ciò non provochi una fuoriuscita della carta dal supporto.

TEST (Comando)

	TI 99/4A
	VIC 20
	CBM 64
	ZX 81
	ZX SPECTRUM
X	MZ-700
	IBM P.C.
	OLIVETTI M20
	APPLE

Fa stampare dei quadrati nei quattro colori per controllare il funzionamento del plotter e delle penne (Modo "Testo").

Formalismo sintattico:

Esempi: in modo immediato:
TEST
Vengono stampati quattro quadrati nei diversi colori

In modo differito:

```
10 MODE TN
20 TEST
30 SKIP 5
40 LIST/P
```

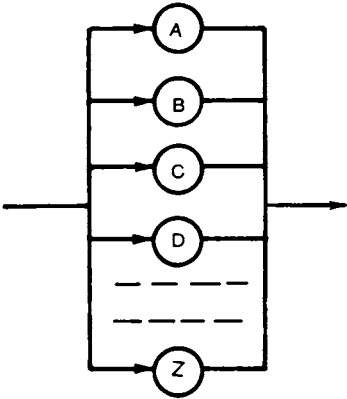
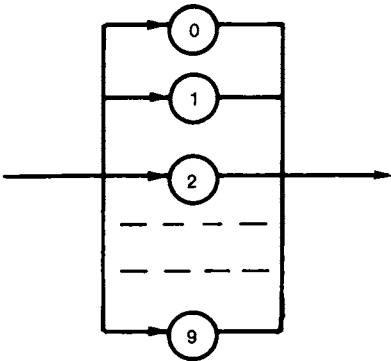
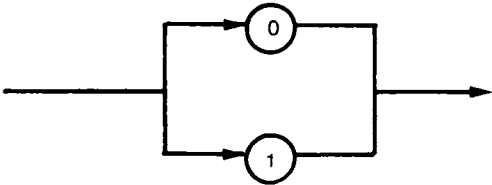
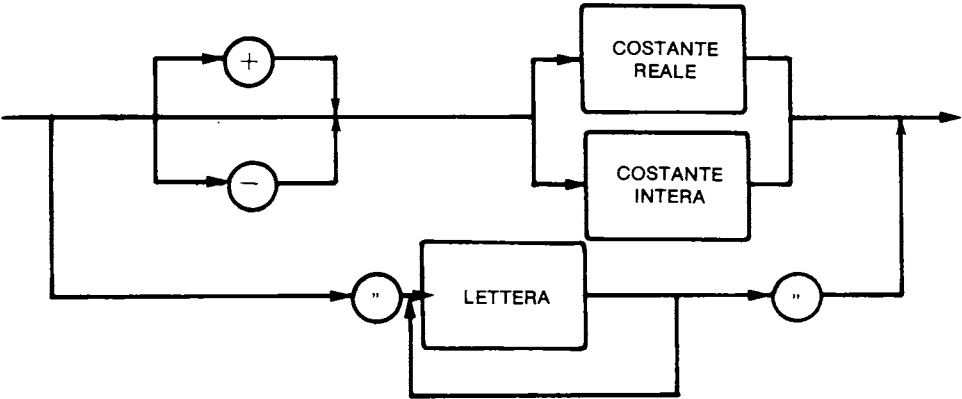
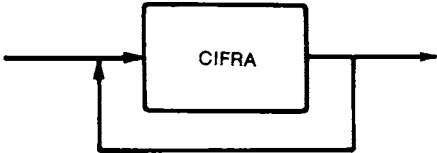
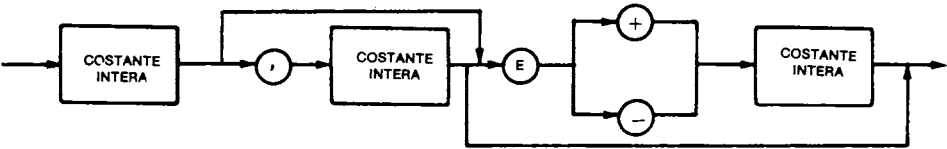
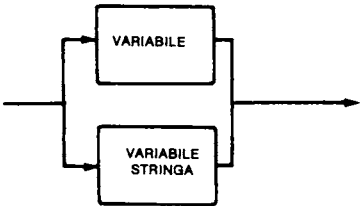
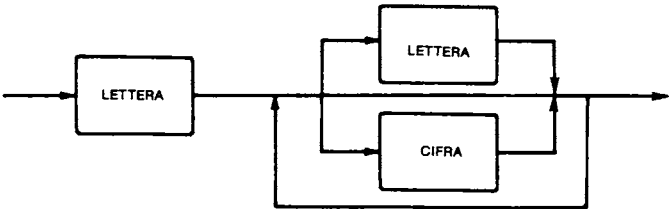
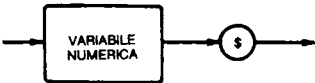
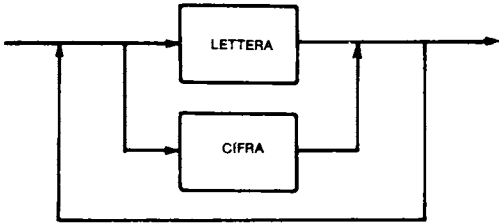
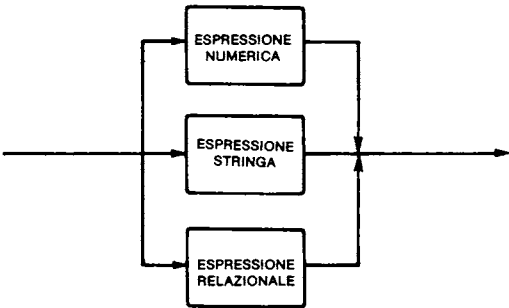
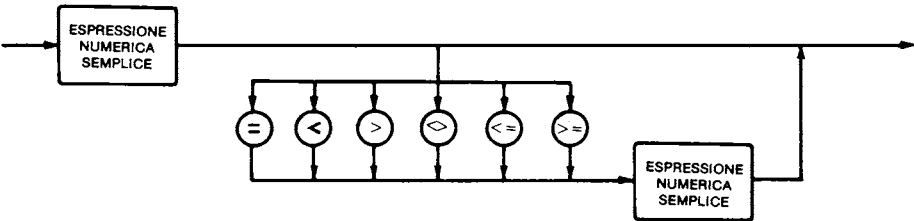
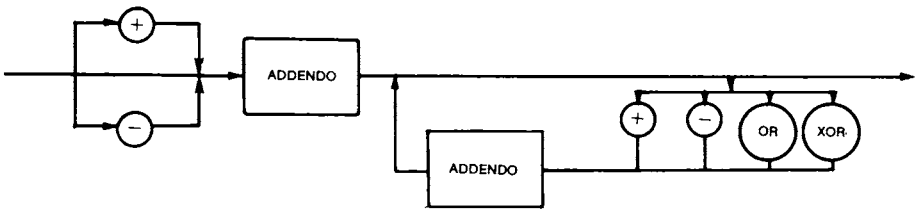
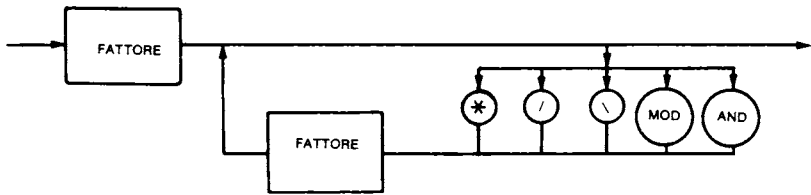
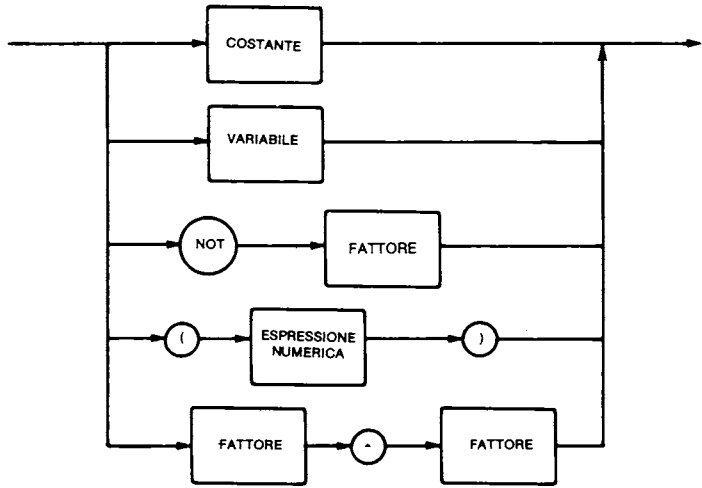
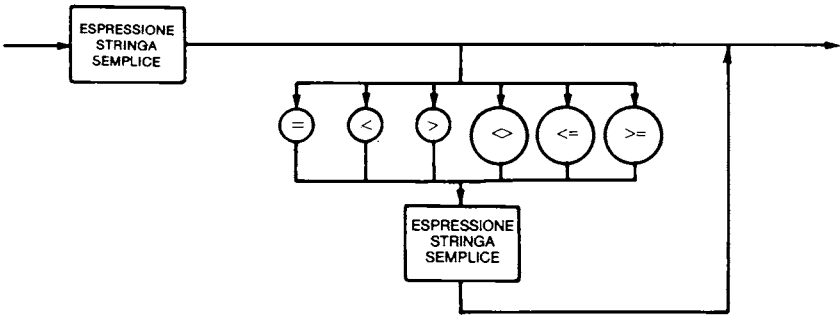
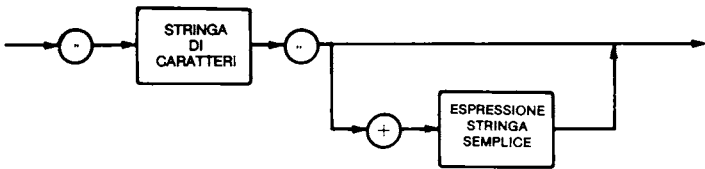
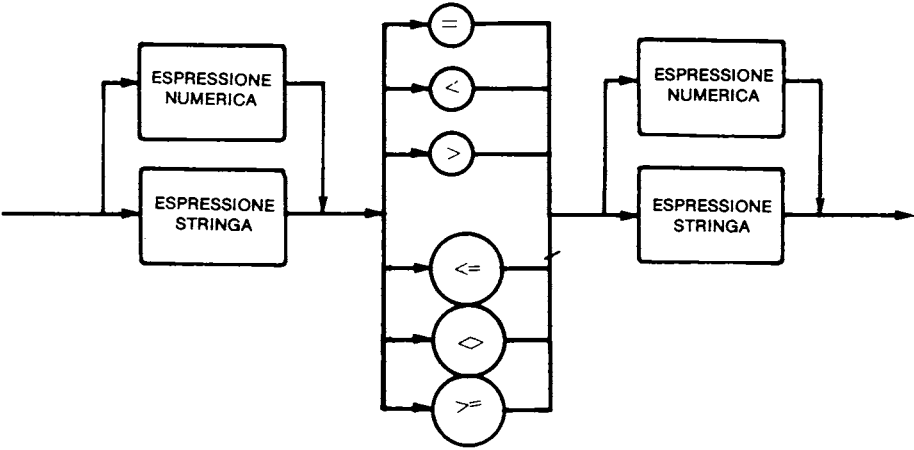
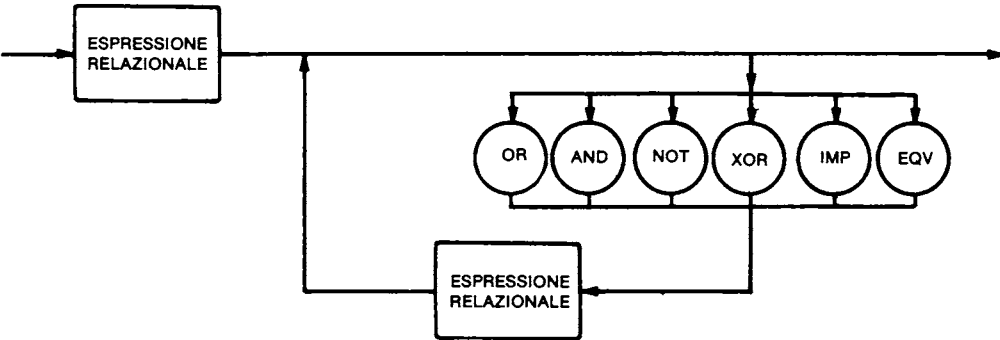
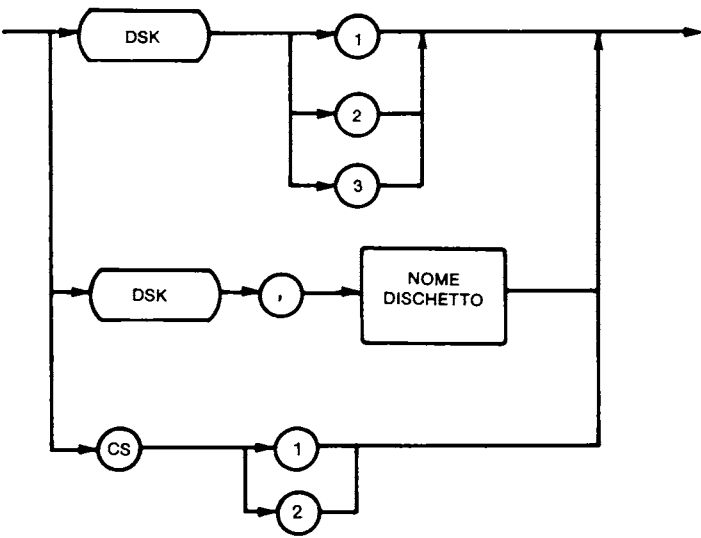
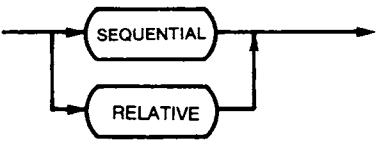
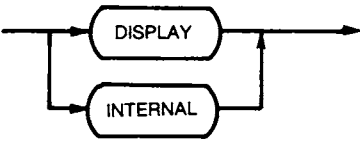
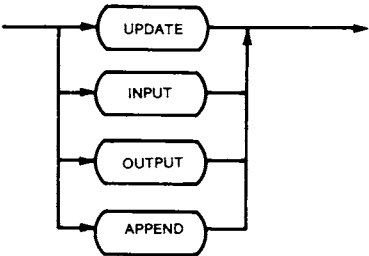
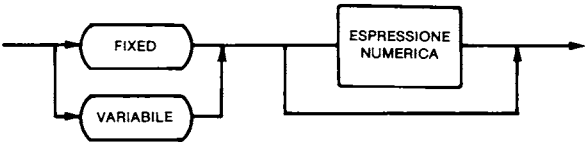
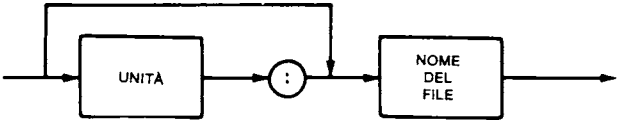
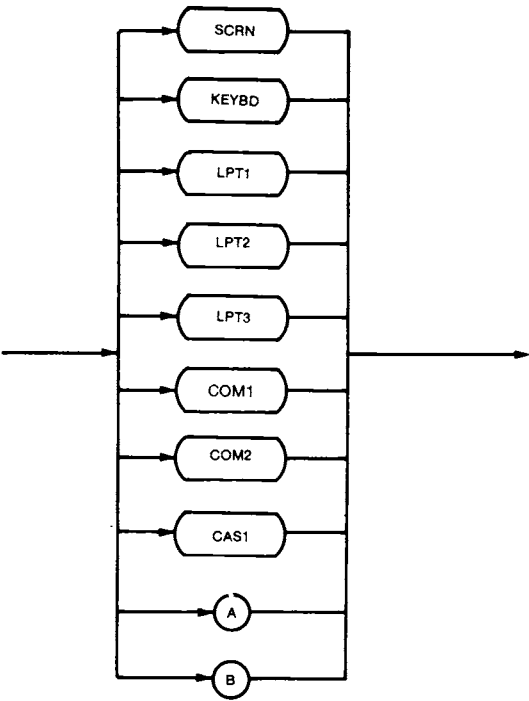
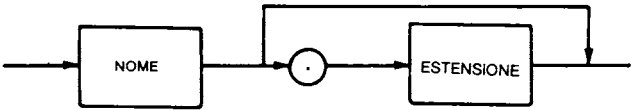
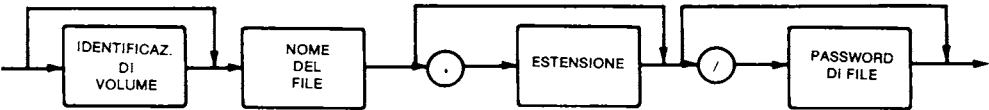
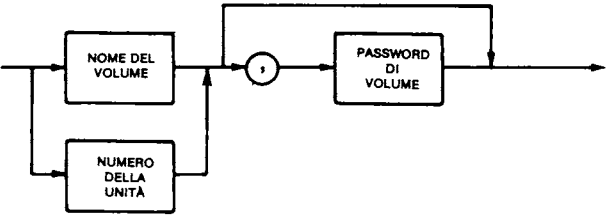
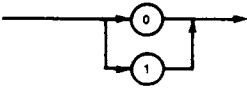
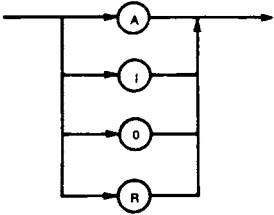
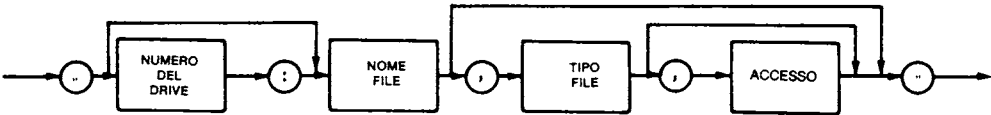
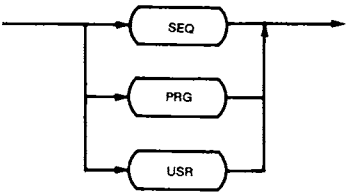
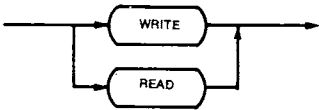
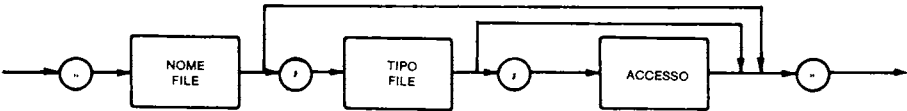
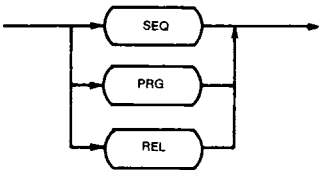
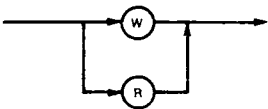
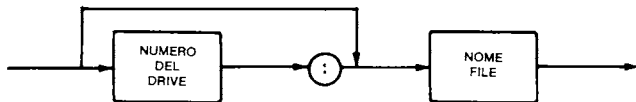
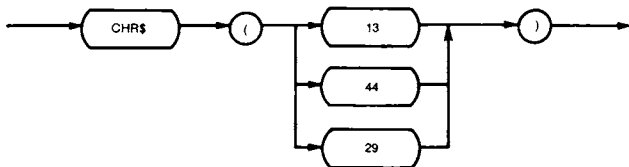
Risultato:
LINEA 10: Commuta la stampante-plotter al modo operativo testo
LINEA 20: Stampa i quattro quadrati nei diversi colori
LINEA 30: Fa avanzare la carta di 5 righe
LINEA 40: Emette la lista del programma sulla stampante-plotter

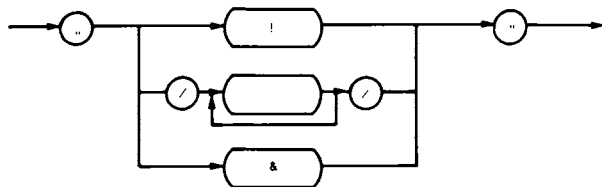
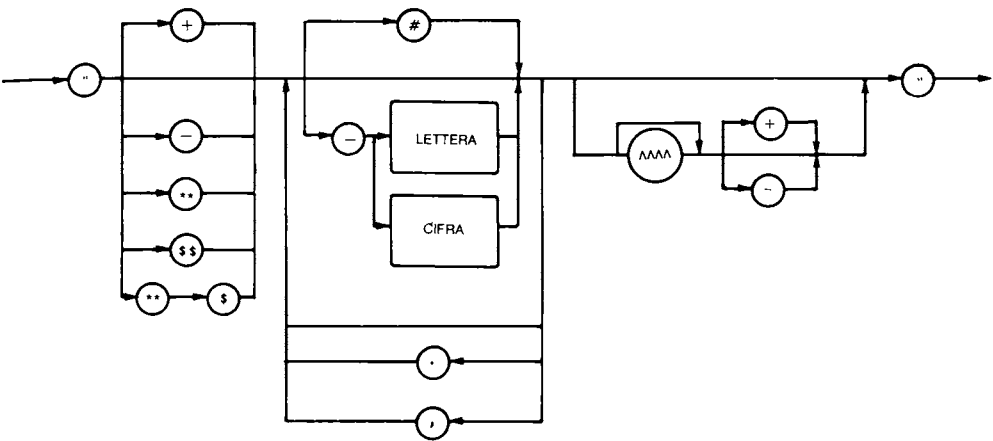
Note:
— Se i pennini sono stati inseriti in modo corretto, la configurazione dei quadrati che viene stampata all'esecuzione di questo comando è la seguente:

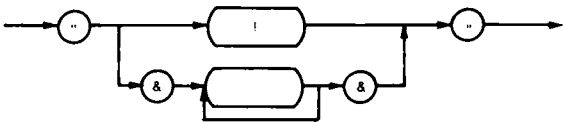
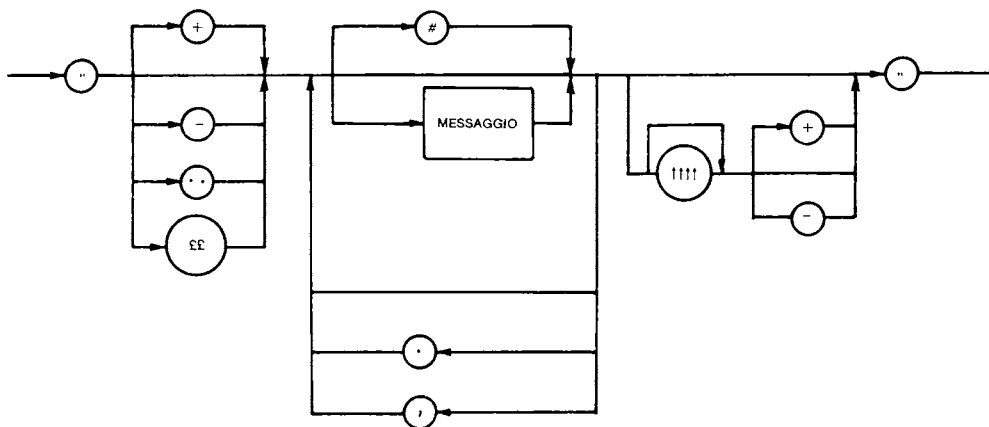
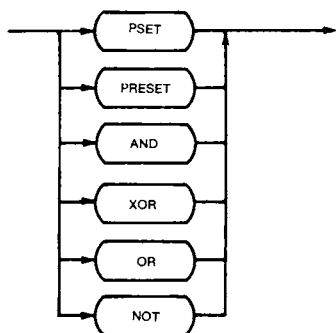
□	□	□	□
0	1	2	3
(nero)	(blu)	(verde)	(rosso)

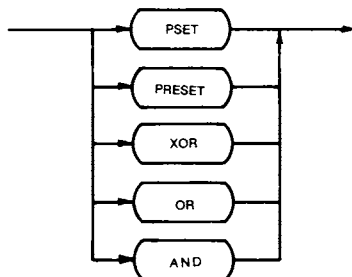
APPENDICE A

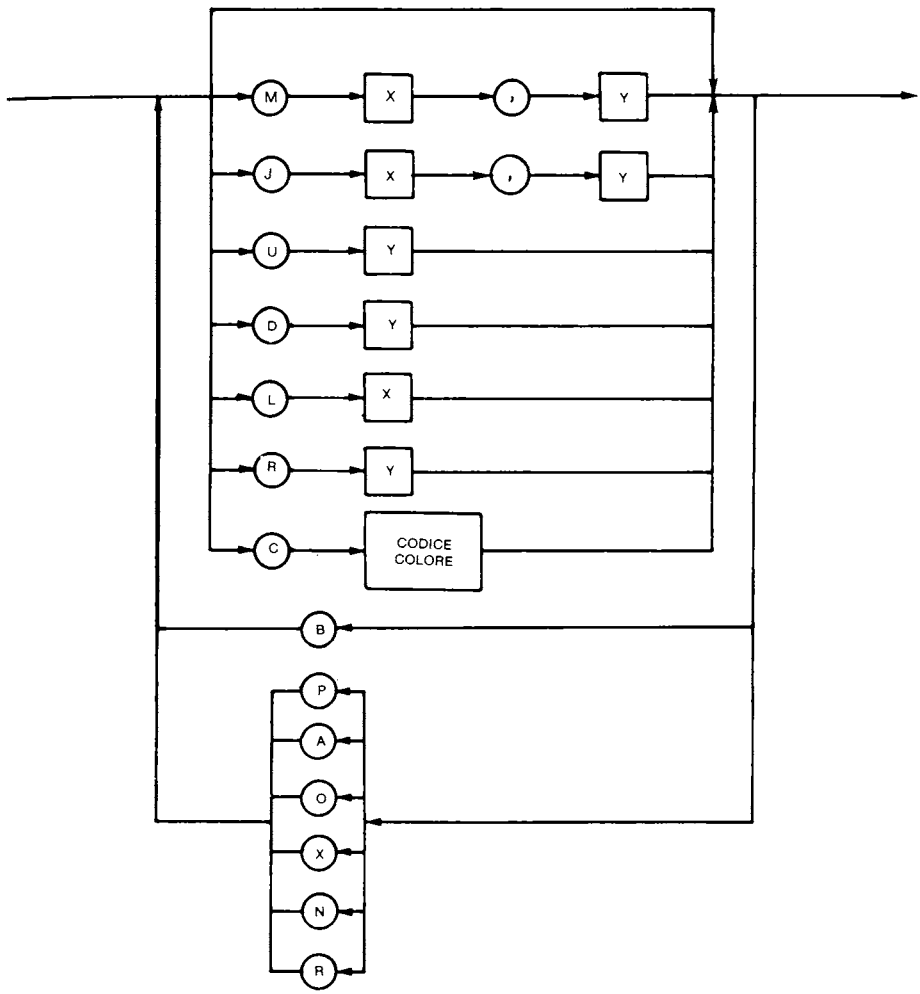
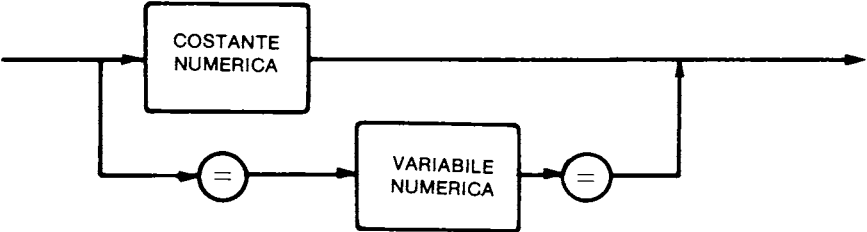
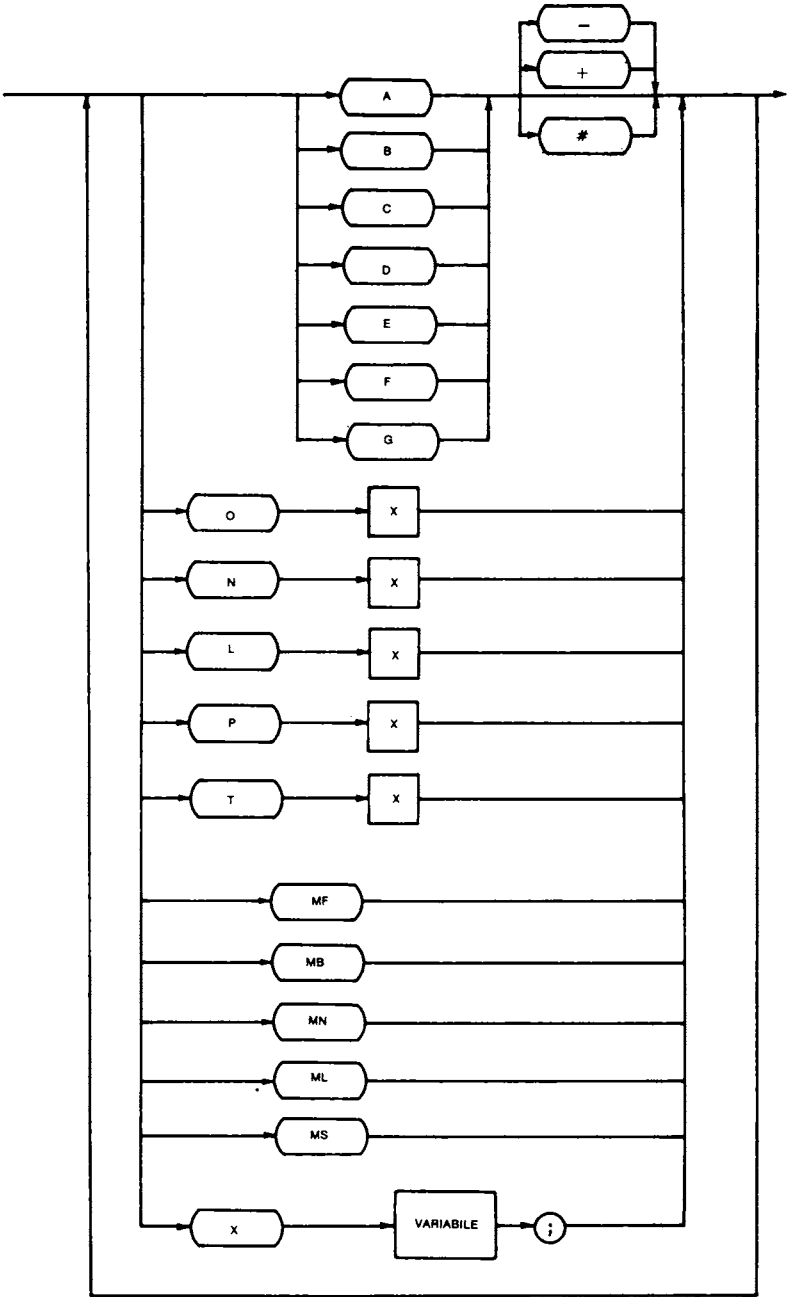
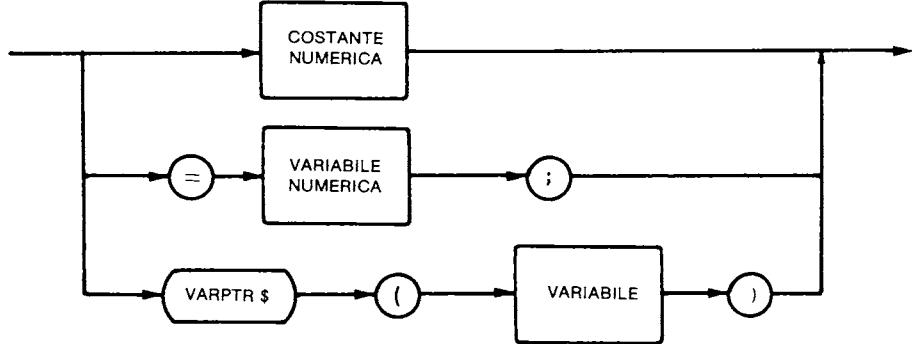
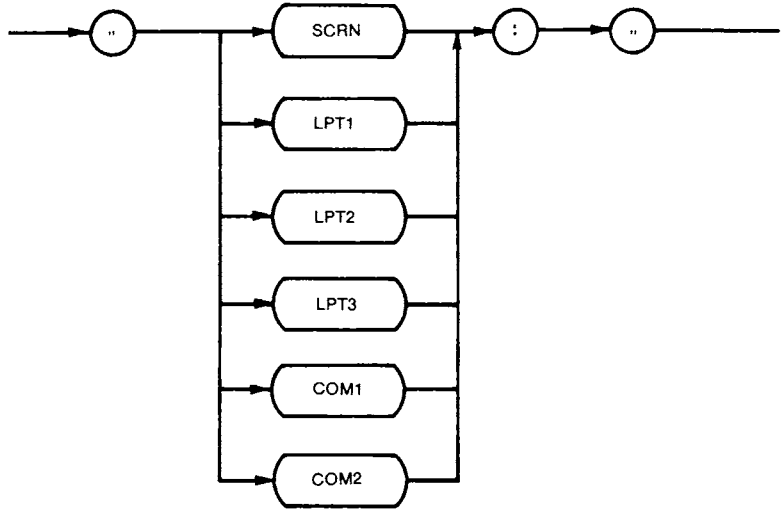
FORMALISMO SINTATTICO DEGLI ELEMENTI NON TERMINALI

LETTERA

CIFRA

DIGIT

COSTANTE

COSTANTE INTERA

COSTANTE REALE

VARIABILE

VARIABILE NUMERICA

VARIABILE STRINGA

STRINGA DI CARATTERI

ESPRESSIONE

ESPRESSIONE NUMERICA

ESPRESSIONE NUMERICA SEMPLICE

ADDENDO

FATTORE

ESPRESSIONE STRINGA

ESPRESSIONE STRINGA SEMPLICE

ESPRESSIONE RELAZIONALE

CONDIZIONE

NOME DISPOSITIVO

ORGANIZZAZIONE FILE (NELLA OPEN) PER: TI99/4A

TIPO FILE (NELLA OPEN) PER: TI99/4A

MODO DI APERTURA (NELLA OPEN) PER: TI99/4A

TIPO RECORD (NELLA OPEN) PER: TI99/4A

IDENTIFICATORE DI FILE
PER: I.B.M. P.C.

DOVE: < UNITA' > È DATA DA:

E: < NOME DEL FILE > È DATO DA:

IDENTIFICATORE DI FILE
PER: OLIVETTI M20

DOVE: < IDENTIFICATORE DI VOLUME >

E: < NUMERO DELL'UNITA' >

METODO DI ACCESSO (NELLA OPEN) PER: OLIVETTI M20

IDENTIFICATORE DEL FILE (IN OPEN)
PER: VIC 20

DOVE: < TIPO FILE >

E: < ACCESSO >

IDENTIFICATORE DEL FILE (IN OPEN)
PER: CBM 64

DOVE: < TIPO FILE >

E: < ACCESSO >

NOME FILE (IN LOAD, SAVE E VERIFY) PER: VIC 20
CBM 64

RITORNO A CAPO (IN PRINT #) PER: VIC 20
CBM 64

STRINGA FORMATO (PER: I.B.M. P.C.
OLIVETTI M20)

1) PER STRINGHE

2) PER CAMPI NUMERICI

STRINGA FORMATO (PER: MZ-700)

1) PER STRINGHE

2) PER CAMPI NUMERICI

PARAMETRO CIRCLE (PER: OLIVETTI M20)
PARAMETRO LINE
PARAMETRO PUT

PARAMETRO PUT (PER: I.B.M. P.C.)

U
X
D
X
L
X
R
X
E
X
F
X
G
X
H
X
M
+
−
X
,
Y
A
X
C
X
S
X
X
VARIABILE
B
N

DOVE X E Y:

STRINGA COMANDI (NELLA DRAW) PER: OLIVETTI M20

DOVE X E Y:

STRINGA COMANDI (NELLA PLAY) PER: I.B.M. P.C.

DOVE X:

UNITA' (NELLA WIDTH) PER: I.B.M. P.C.

PARITA' (IN OPEN "COM") PER: I.B.M. P.C.

APPENDICE B

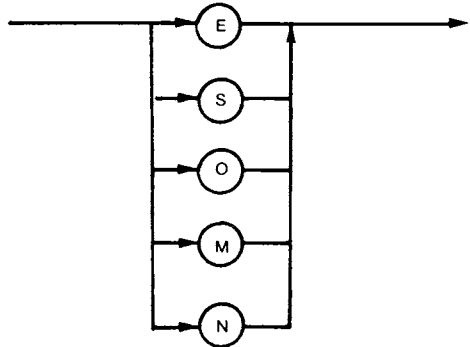



TABELLA CON ELENCO GLOBALE DELLE ISTRUZIONI, COMANDI E FUNZIONI

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Microprocessore (C.P.U.)	TMS9900	6502	6510	Z80A	Z80A	Z80A	8088	Z8001	6502
Versione di BASIC/Sistema Operativo (eventuale)	TI-BASIC	CBM-BASIC	CBM-BASIC	BASIC	BASIC	S-BASIC	BASIC 1.10/ MS-DOS	MICROSOFT/ PCOS	APPLESOFT/ DOS 3.3
OPERATORI LOGICI									
AND	*	AND	AND	AND	AND	*	AND	AND	AND
OR	+	OR	OR	OR	OR	+	OR	OR	OR
NOT	—	NOT	NOT	NOT	NOT	—	NOT	NOT	NOT
XOR	—	—	—	—	—	—	XOR	XOR	—
IMP	—	—	—	—	—	—	IMP	IMP	—
EQV	—	—	—	—	—	—	EQV	EQV	—
VARIABILI	—	—	—	—	—	—	DEFINT DEFSNG DEFDBL DEFSTR	DEFINT DEFSNG DEFDBL DEFSTR	—
OPERATORI PARTICOLARI									
Divisione per intero	—	—	—	—	—	—	\	\	—
MOD (Aritmetica modulare)	—	—	—	—	—	—	MOD	MOD	—
Concatenamento di stringhe	&	+	+	+	+	+	+	+	+
MATRICI									
Dimensionamento di matrici	DIM	DIM	DIM	DIM	DIM	DIM	DIM	DIM	DIM
Valore minimo degli indici	OPTION BASE	—	—	—	—	—	OPTION BASE	OPTION BASE	—
Elimina le matrici dal programma	—	—	—	—	—	—	ERASE	ERASE	—

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
COMANDI E ISTRUZIONI PIU' COMUNI a) COMANDI Cancellazione del programma	NEW	NEW	NEW	NEW	NEW	NEW	NEW	NEW	NEW
Lista del programma a video	LIST	LIST	LIST	LIST	LIST	LIST	LIST	LIST	LIST
Esecuzione del programma	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN
Generazione automatica dei numeri di linea	NUMBER	—	—	—	—	AUTO	AUTO	AUTO	—
Rinumerazione dei numeri di linea	RESEQUENCE	—	—	—	—	RENUM	RENUM	RENUM	—
Esecuzione con tracciamento	TRACE	—	—	—	—	TRON	TRON	TRON	TRACE
Fa terminare l'esecuzione con tracciamento	UNTRACE	—	—	—	—	TROFF	TROFF	TROFF	NOTRACE
Inserzione di punti di arresto nel programma	BREAK	—	—	—	—	—	—	—	—
Eliminazione dei punti di arresto	UNBREAK	—	—	—	—	—	—	—	—
Proseguimento dopo un arresto	CONTINUE	CONT	CONT	CONT	CONTINUE	CONT	CONT	CONT	CONT
Salvataggio del programma BASIC	SAVE	SAVE	SAVE	SAVE	SAVE	SAVE	SAVE	SAVE	SAVE
Salvataggio del programma in linguaggio macchina	—	—	—	—	SAVE...CODE	—	BSAVE	—	BSAVE[1]
Memorizza una matrice su supporto esterno	—	—	—	—	SAVE...DATA	—	—	—	STORE
Verifica dei programmi memorizzati su nastro	X	VERIFY	VERIFY	—	VERIFY	VERIFY	—	—	—
Caricamento di un programma BASIC da supporto esterno	OLD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD
Caricamento di un programma in linguaggio macchina da supporto esterno	—	LOAD	LOAD	—	LOAD	LOAD	BLOAD	—	BLOAD[1]

[1] Comandi del DOS 3.3

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Richiama una matrice da supporto esterno	—	—	—	—	LOAD...DATA	—	—	—	RECALL
b) ISTRUZIONI DI ASSEGNAMENTO									
Assegnazione di valori a variabili	LET	LET	LET	LET	LET	LET	LET	LET	LET
Azzeramento variabili	—	CLR	CLR	CLEAR	CLEAR	CLR	CLEAR	CLEAR	CLEAR
c) ISTRUZIONI CONDIZIONALI									
Salto condizionato - Norma	IF...THEN... ELSE...	IF...THEN...	IF...THEN...	IF...THEN...	IF...THEN...	IF...THEN...	IF...THEN... ELSE...	IF...THEN... ELSE...	IF...THEN...
Salto condizionato - Alternativa 1	—	—	IF...GOTO...	—	—	IF...GOTO...	IF...GOTO... ELSE...	IF...GOTO... ELSE...	IF...GOTO...
Salto condizionato - Alternativa 2	—	—	—	—	—	IF...GOSUB...	—	—	—
d) ISTRUZIONI DI INPUT/OUTPUT									
Stampa dati sullo schermo video con un formato standard	PRINT	PRINT	PRINT	PRINT	PRINT	PRINT	PRINT WRITE	PRINT WRITE	PRINT
Ingresso dati da tastiera	INPUT	INPUT	INPUT	INPUT	INPUT LINE INPUT	INPUT	INPUT LINE INPUT	INPUT LINE INPUT	INPUT
Acquisizione caratteri da tastiera	CALL KEY	GET	GET	INKEY$	INKEY$	GET	INKEY$	INKEY$	GET
Stampa dati sullo schermo video con un formato definito dall'utente	—	—	—	—	—	PRINT USING	PRINT USING	PRINT USING	—
Creazione di un file dati interno	DATA	DATA	DATA	—	DATA	DATA	DATA	DATA	DATA
Lettura del file dati interno	READ	READ	READ	—	READ	READ	READ	READ	READ
Risistemazione dei puntamenti alle istruzioni DATA	RESTORE	RESTORE	RESTORE	—	RESTORE	RESTORE	RESTORE	RESTORE	RESTORE

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
a) ISTRUZIONI CHE MODIFICANO LA SEQUENZA DI ESECUZIONE									
Salto incondizionato	GOTO	GOTO	GOTO	GOTO	GOTO	GOTO	GOTO	GOTO	GOTO
Arresto del programma	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP
Fine del programma	END	END	END	—	—	END	END	END	END
Richiamo di "subroutine"	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB
Uscita da "subroutine"	RETURN	RETURN	RETURN	RETURN	RETURN	RETURN	RETURN	RETURN	RETURN POP
Salto multiplo	ON..GOTO...	ON..GOTO...	ON..GOTO...	—	—	ON..GOTO...	ON..GOTO...	ON..GOTO...	ON..GOTO...
Richiamo di "subroutine" multiplo	ON..GOSUB...	ON..GOSUB...	ON..GOSUB...	—	—	ON..GOSUB...	ON..GOSUB...	ON..GOSUB...	ON..GOSUB...
Iterazione (o "Loop")	FOR	FOR	FOR	FOR	FOR	FOR	FOR	FOR	FOR
Fine di una iterazione	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT
f) VARIE									
Commenti	REM	REM	REM	REM	REM	REM	REM	REM	REM
Origine della sequenza di numeri casuali	RANDOMIZE	—	—	RAND	RANDOMIZE	—	RANDOMIZE	RANDOMIZE	—
Generazione numeri casuali	RND	RND	RND	RND	RND	RND	RND	RND	RND
Modifica della memoria	—	POKE	POKE	POKE	POKE	POKE	POKE	—	POKE

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
RITORNO AL BASIC									
Interruzione del programma	tasto BREAK	tasto RUN/STOP	tasto RUN/STOP	tasto BREAK	tasto BREAK	tasti SHIFT+BREAK	tasti CTRL+BREAK	tasti CTRL+C	tasti CTRL+C
Ritorno al BASIC	tasto QUIT	tasti RUN/STOP +RESTORE	tasti RUN/STOP +RESTORE	—	—	tasti CTRL +RESET	tasti CTRL+BREAK	tasti SHIFT+RESET	tasto RESET
Ritorno al sistema operativo (o al Monitor del BASIC)	BYE	—	—	—	—	BYE	SYSTEM	SYSTEM	CALL-151
CANCELLAZIONE VIDEO									
Cancellazione video in modo immediato	CALL CLEAR	tasto CLR	tasto CLR	CLS	CLS	tasto CLR	tasti CTRL+HOME	CLS	tasti ESC+@
Cancellazione video in modo differito	CALL CLEAR	PRINT " "	PRINT " "	CLS	CLS	PRINT " "	CLS	CLS	HOME CALL-936
CORREZIONI A UN PROGRAMMA									
Richiama la linea di programma da modificare	EDIT	—	—	EDIT	EDIT	—	EDIT	EDIT	—
Cancellazione righe di programma	—	—	—	—	—	DELETE	DELETE	DELETE	DEL
POSIZIONAMENTO DEL CURSORE									
Posizionamento del cursore in modo programma	—	—	—	PRINT...AT...	PRINT...AT...	CURSOR	LOCATE	CURSOR CURSOR POINT	HTAB VTAB

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
FUNZIONI									
a) FUNZIONI NUMERICHE									
Valore assoluto	ABS	ABS	ABS	ABS	ABS	ABS	ABS	ABS	ABS
Valore dell'arcocoseno	—	—	—	ACS	ASC	—	—	—	—
Valore ASCII di un carattere	ASC	ASC	ASC	CODE	CODE	ASC	ASC	ASC	ASC
Valore dell'arcoseno	—	—	—	ASN	ASN	—	—	—	—
Valore dell'arcotangente	ATN	ATN	ATN	ATN	ATN	ATN	ATN	ATN	ATN
Conversione in un numero a precisione doppia	—	—	—	—	—	—	CDBL	CDBL	—
Conversione in un numero intero	—	—	—	—	—	—	CINT	CINT	—
Conversione in un numero a precisione singola	—	—	—	—	—	—	CSNG	CSNG	—
Valore del coseno	COS	COS	COS	COS	COS	COS	COS	COS	COS
Conversione di una stringa in un numero (intero, a singola o a doppia precisione)	—	—	—	—	—	—	CVI CVS CVD	CVI CVS CVD	—
Esponenziale (potenza di "e")	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP
Tronca un'espressione a un valore intero	—	—	—	—	—	—	FIX	FIX	—
Quantità di memoria utilizzabile (in numero di byte)	—	FRE	FRE	—	—	SIZE	FRE	FRE	FRE
Posizione relativa (tra stringhe)	POS	—	—	—	—	—	INSTR	INSTR	—
Valore intero di un'espressione	INT	INT	INT	INT	INT	INT	INT	INT	INT
Lunghezza di una stringa	LEN	LEN	LEN	LEN	LEN	LEN	LEN	LEN	LEN
Logaritmo naturale	LOG	LOG	LOG	LN	LN	LN	LOG	LOG	LOG

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Logaritmo decimale	—	—	—	—	—	LOG	—	—	—
Lettura di un byte dalla memoria	—	PEEK	PEEK	PEEK	PEEK	PEEK	PEEK	—	PEEK
Valore di π (PI Greco)	—	tasto π (CHR$(126))	tasto π (CHR$(126))	PI	PI	PAI	—	—	—
Ritorna la posizione del cursore	—	POS	POS	—	—	—	POS CSRLIN	POS	POS
Conversione gradi → radianti	—	—	—	—	—	RAD	—	—	—
Ritorna il segno di una espressione	SGN	SGN	SGN	SGN	SGN	SGN	SGN	SGN	SGN
Valore del seno	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN
Inserimento di spazi in stampa	—	SPC	SPC	—	—	SPC	SPC	SPC	SPC
Radice quadrata	SQR	SQR	SQR	SQR	SQR	SQR	SQR	SQR	SQR
Tabulazione (con la PRINT)	TAB	TAB	TAB	TAB	TAB	TAB	TAB	TAB	TAB
Valore della tangente	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN
Conversione di una stringa in un valore numerico	VAL	VAL	VAL	VAL	VAL	VAL	VAL	VAL	VAL
b) FUNZIONI DI STRINGA									
Ritorna il carattere corrispondente ad un codice ASCII voluto	CHR$	CHR$	CHR$	CHR$	CHR$	CHR$	CHR$	CHR$	CHR$
Conversione decimale → esadecimale	—	—	—	—	—	—	HEX$	HEX$	—
Conversine decimale → ottale	—	—	—	—	—	—	OCT$	OCT$	—
Estrazione di sottostringhe	SEG$	LEFT$ MID$ RIGHT$	LEFT$ MID$ RIGHT$	TO	TO	LEFT$ MID$ RIGHT$	LEFT$ MID$ RIGHT$	LEFT$ MID$ RIGHT$	LEFT$ MID$ RIGHT$
Manipolazione di stringhe	—	—	—	—	—	—	STRING$	STRING$	—

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Conversione numero → stringa	STR$	STR$	STR$	STR$	STR$	STR$	STR$	STR$	STR$
Conversione di un numero intero, o a precisione singola o doppia, in una stringa	—	—	—	—	—	—	MKI$ MKS$ MKD$	MKI$ MKS$ MKD$	—
Ritorna una stringa a spazi	—	—	—	—	—	—	SPACE$	SPACE$	—
Conversione di una stringa in una stringa	—	—	—	—	VAL$	—	—	—	—
VARIABILI DI SISTEMA									
a) VARIABILI NUMERICHE									
Numero di linea in cui è stato intercettato l'errore	—	—	—	—	—	ERN	ERL	ERL	—
Codice dell'errore	—	—	—	—	—	ERR	ERR	ERR	—
Velocità di immissione/emissione di caratteri	—	—	—	—	—	—	—	—	SPEED
Stato dell'ultima operazione di I/O	—	ST	ST	—	—	—	—	—	—
Tempo dall'accensione (come valore numerico)	—	TI	TI	—	—	—	—	—	—
b) VARIABILI STRINGA									
Contiene la data	—	—	—	—	—	—	DATE$	DATE$	—
Tempo dall'accensione (come stringa)	—	TI$	TI$	—	—	TI$	TIME$	TIME$	—
FUNZIONI DEFINITE DALL'UTENTE									
Definizione di funzioni da parte dell'utente	DEF	DEF FN	DEF FN	—	DEF FN	DEF FN	DEF FN	DEF FN	DEF FN
Richiamo di funzioni definite dall'utente	X	FN	FN	—	FN	FN	FN	FN	FN

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
GRAFICA E COLORE									
a) ISTRUZIONI DI CARATTERE GENERALE									
Ritorna gli attributi di una posizione dello schermo	—	—	—	—	ATTR	—	—	—	SCRN
Tracciamento di un cerchio su video	—	—	—	—	CIRCLE	—	CIRCLE	CIRCLE	—
Visualizzazione caratteri a video	CALL HCHAR CALL VCHAR	(2)	(2)	(3)	(3)	(2)	(4)	(4)	—
Acquisizione caratteri da video	CALL GCHAR	(5)	(5)	—	SCREEN$	(5)	SCREEN	—	—
Colora un carattere o un gruppo di caratteri, ed eventualmente li fa lampeggiare	CALL COLOR	(2)	(2)	—	INK PAPER BRIGHT FLASH	COLOR PRINT [ ]	—	—	COLOR HCOLOR
Colora lo schermo	CALL SCREEN	(2)	(2)	—	INK PAPER BRIGHT FLASH	COLOR	COLOR	COLOR	COLOR HCOLOR
Lampeggiamento dell'intero video	—	—	—	—	—	—	—	—	FLASH
Colora il margine dello schermo	—	(2)	(2)	—	BORDER	—	COLOR	—	—
Definizione dell'area di scorrimento	—	—	—	—	—	CONSOLE	—	—	—
Traccia linee sullo schermo	—	—	—	—	DRAW	—	DRAW LINE	DRAW LINE	HLIN VLIN
Disegna profili sullo schermo	—	—	—	—	—	—	—	—	DRAW XDRAW
Memorizza dati da video in una matrice	—	—	—	—	—	—	GET	GET	—
Richiama dati a video da una matrice	—	—	—	—	—	—	PUT	PUT	—

(2) È possibile, con delle POKE nella memoria schermo
(4) È possibile con WRITE
(3) È possibile con PRINT..AT..
(5) È possibile con delle PEEK dalla memoria schermo

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Funzionamento in modo grafico a bassa risoluzione	—	—	—	—	—	—	—	—	GR
Passaggio al modo grafico ad alta risoluzione	—	—	—	—	—	—	SCREEN	—	HGR HGR2
Stampa i caratteri in inverso	—	tasto RVS/ON	tasto RVS/ON	—	INVERSE 1	[2]	—	—	INVERSE
Ritorno al modo normale del video dopo una inversione	—	tasto RVS/OFF	tasto RVS/OFF	—	INVERSE 0	[2]	—	—	NORMAL
Stampa un carattere senza cancellare il precedente	—	—	—	—	OVER	—	—	—	—
Attiva i punti luminosi (pixel)	—	—	—	PLOT	PLOT	SET	PAINT PSET PRESET	PAINT PSET PRESET	PLOT HPLOT
Disattiva i punti luminosi	—	—	—	UNPLOT	PLOT OVER 1	RESET	PSET PRESET	PSET PRESET	PLOT
Testa se un punto luminoso è acceso o no	—	—	—	—	POINT	—	POINT	POINT	—
Trasformazione di coordinate	—	—	—	—	—	—	—	SCALE SCALEX SCALEY	—
Passaggio al funzionamento in modo testo	—	—	—	—	—	—	—	—	TEXT
Apertura di una finestra	—	—	—	—	—	—	—	WINDOW	—
Chiusura di una finestra	—	—	—	—	—	—	—	CLOSE WINDOW	—
Larghezza della riga di emissione (numero di colonne, se per lo schermo)	—	—	—	—	—	—	WIDTH	WIDTH	POKE 33,X
b) GENERAZIONE CARATTERI	SI	SI	SI	NO	SI	NO	NO	NO	NO
Generazione caratteri	CALL CHAR	[6]	[6]	—	BIN+USR	—	—	—	—

[2] È possibile con delle POKE nella memoria schermo

[6] È possibile con delle POKE

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
c) GENERAZIONE PROFILI									
Rotazione angolare del profilo	—	—	—	—	—	—	—	—	ROT
Dimensione di un profilo	—	—	—	—	—	—	—	—	SCALE
Caricamento profili da nastro magnetico	—	—	—	—	—	—	—	—	SHLOAD
GENERAZIONE SUONI									
Generazione suoni	CALL SOUND	[6]	[6]	—	BEEP	MUSIC+ TEMPO	BEEP PLAY SOUND	—	—
GESTIONE DEGLI ERRORI									
"Testa" le variabili di sistema per la gestione degli errori	—	—	—	—	—	IF ERN..THEN IF ERL..THEN	IF ERR..THEN IF ERL..THEN	IF ERR..THEN IF ERL..THEN	—
Simula il ripetersi di un errore	—	—	—	—	—	—	ERROR	ERROR	—
Intercettazione degli errori	—	—	—	—	—	ON ERROR GOTO	ON ERROR GOTO	ON ERROR GOTO	ONERR GOTO
Ripartenza del programma	—	—	—	—	—	RESUME	RESUME	RESUME	RESUME
GESTIONE DEI FILE									
Chiusura di un file	CLOSE	CLOSE	CLOSE	—	CLOSE[8]	CLOSE	CLOSE	CLOSE	CLOSE[1]
Fine del file	EOF	ST=64	ST=64	—	—	—	EOF	EOF	—
Allocazione spazio per un file ad accesso diretto (RANDOM)	—	—	—	—	—	—	FIELD	FIELD	MAXFILES[1]
Lettura da un file sequenziale	INPUT#	INPUT#	INPUT#	—	INPUT#[8] INKEY$#[8]	INPUT/T	INPUT# LINE INPUT#	INPUT# LINE INPUT#	—
Formatta le cartucce del Microdrive	—	—	—	—	FORMAT[8]	—	—	—	—

[1] Comandi del DOS 3.3
[8] Non funziona senza microdrive
[6] È possibile con delle POKE

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Lettura da un file ad accesso diretto (RANDOM)	—	GET#	GET#	—	—	—	GET#	GET#	—
Lettura da una stringa da una unità a disco	—	—	—	—	—	—	INPUT$	INPUT$	—
Scrittura dati su un file sequenziale * secondo un formato standard	PRINT#	PRINT#	PRINT#	—	PRINT#[8]	PRINT/T	PRINT# WRITE#	PRINT# WRITE#	PRINT[1] WRITE[1]
Scrittura dati su un file sequenziale con formato definito dall'utente	—	—	—	—	—	—	PRINT#USING	PRINT#USING	—
Scrittura dati su un file ad accesso diretto (RANDOM)	—	—	—	—	—	—	PUT#	PUT#	—
Ritorna la posizione attuale nel file	—	—	—	—	—	—	LOC	LOC	—
Lunghezza del file	—	—	—	—	—	—	LOF	LOF	—
Sposta i dati nella memoria di transito per file ad accesso diretto (RANDOM)	—	—	—	—	—	—	LSET RSET	LSET RSET	—
Apertura di un file	OPEN	OPEN	OPEN	—	OPEN[8]	ROPEN WOPEN	OPEN	OPEN	OPEN[1]
Chiude tutti i file su minidisco	—	—	—	—	—	—	RESET	—	—
Riposizionamento all'inizio del file	RESTORE#	—	—	—	—	—	—	—	—
Preleva i dati da un file sorgente e li invia ad un file di destinazione	—	—	—	—	MOVE[8]	—	—	—	—

[1] Comandi del DOS 3.3

[8] Non funziona senza microdrive

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
COMANDI E ISTRUZIONI PARTICOLARI									
Concatenamento di programmi	—	—	—	—	—	—	CHAIN	CHAIN	CHAIN[1]
Trasferimento output da schermo a dispositivo esterno	—	CMD	CMD	—	—	—	—	—	—
Acquisizione dati da barre comando giochi (Joystick o Paddle)	CALL JOYST	[7]	[7]	—	—	JOY	STICK	—	PDL
Attiva l'adattatore per le comunicazioni	—	—	—	—	—	—	COM	—	—
Passaggio variabili a un programma concatenato	—	—	—	—	—	—	COMMON	COMMON	—
Stampa un'immagine dello schermo (Hard-copy) su stampante	—	—	—	COPY	COPY	—	tasto PrtSc	—	—
Definizione tasti funzionali	—	—	—	—	—	DEF KEY	KEY KEY ON KEY OFF	—	—
Definizione segmento di memoria	—	—	—	—	—	—	DEF SEG	—	—
Cancellazione di un file su disco	DELETE	—	—	—	ERASE[8]	—	KILL	KILL	DELETE[1]
Elaborazione in modo veloce	—	—	—	FAST	—	—	—	—	—
Elenco file su minidisco	—	—	—	—	CAT[8]	—	FILES	FILES	CATALOG[1]
Numero di files che possono venir aperti contemporaneamente	CALL FILES	—	—	—	—	—	—	—	—
Trasferimento dati dalle porte di I/O	—	GET#	GET#	—	IN	INP	INP	—	IN#
Lista dei tasti funzionali	—	—	—	—	—	KEY LIST	KEY LIST	—	—

[1] Comandi del DOS 3.3
[8] Non funziona senza microdrive
[7] È possibile con delle PEEK dagli indirizzi opportuni

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Limita l'area di memoria per il programma BASIC	—	SYS	SYS	CLEAR	CLEAR	LIMIT	DEF USR	DEF USR	HIMEM: LOMEM:
Lista di un programma su stampante	LIST	LIST (dopo CMD)	LIST (dopo CMD)	LLIST	LLIST	LIST/P	LLIST	LLIST	LLIST[1]
Ritorna la posizione della testina di stampa	—	—	—	—	—	—	LPOS	LOPS	—
Fusione di programmi	—	—	—	—	MERGE	MERGE	MERGE	MERGE	—
Accende e spegne il motore del registratore a cassetta	—	—	—	—	—	—	MOTOR	—	—
Modifica il nome di un file su minidisco	—	X	X	—	—	—	NAME	NAME	RENAME[1]
Numero di spazi da stampare alla fine della linea	—	—	—	—	—	—	—	NULL	—
Richiamo di "subroutines" in caso di informazioni in arrivo sul canale comunicazioni	—	—	—	—	—	—	ON COM (n) GOSUB..	—	—
Richiamo "subroutines" mediante tasti funzionali	—	—	—	—	—	—	ON KEY (n) GOSUB..	—	—
Richiamo "subroutines" mediante penna luminosa	—	—	—	—	—	—	ON PEN GOSUB..	—	—
Richiamo "subroutines" mediante barra comando per giochi	—	—	—	—	—	—	ON STRIG(n) GOSUB..	—	—
Apertura di un file per le comunicazioni	—	—	—	—	—	—	OPEN "COM	—	—
Trasferimento dati alle porte di I/O	—	PRINT#	PRINT#	—	OUT	OUT	OUT	—	PR#

[1] Comandi del DOS 3.3

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
Legge la penna luminosa	—	—	—	—	—	—	PEN ON PEN OFF PEN STOP PEN (X)	—	—
Stampa su stampante con un formato standard	PRINT#	PRINT (dopo CMD)	PRINT (dopo CMD)	LPRINT	LPRINT	PRINT/P	LPRINT	LPRINT	PRINT (dopo PR#)
Stampa su stampante mediante un formato definito dall'utente	—	—	—	—	—	PRINT/P USING	LPRINT USING	LPRINT USING	—
Elaborazione in modo lento	—	—	—	SLOW	—	—	—	—	—
Stato dei tasti nelle barre comando dei giochi	—	—	—	—	—	—	STRIG ON STRIG OFF STRIG (n)	—	—
Scambia i valori di due variabili	—	—	—	—	—	—	SWAP	SWAP	—
Richiamo programmi in linguaggio macchina	—	USR	USR	USR	USR	USR	USR CALL	USR CALL EXEC	USR CALL
Ritorna l'indirizzo della variabile	—	—	—	—	—	—	VARPTR VARPTR$	VARPTR	—
Interruzione temporanea del programma	—	WAIT	WAIT	PAUSE	PAUSE	—	WAIT	—	WAIT
Esegue un "loop" finchè una determinata condizione è vera	—	—	—	—	—	—	WHILE	WHILE	—
Fine del "loop" iniziato con WHILE	—	—	—	—	—	—	WEND	WEND	—
Fa scorrere lo schermo di una linea verso l'alto	—	—	—	SCROLL	—	—	—	—	—

DESCRIZIONE	TEXAS TI 99/4A	COMMODORE VIC 20	COMMODORE 64	SINCLAIR ZX 81	SINCLAIR SPECTRUM	SHARP MZ 700	IBM P.C.	OLIVETTI M 20	APPLE
COMANDI E ISTRUZIONI PER LA STAMPANTE-PLOTTER DELLO SHARP MZ-731									
Tracciamento assi cartesiani su plotter	—	—	—	—	—	AXIS	—	—	—
Tracciamento di un cerchio su plotter	—	—	—	—	—	CIRCLE	—	—	—
Stampa caratteri su plotter (nel modo grafico)	—	—	—	—	—	GPRINT	—	—	—
Nuova origine delle coordinate	—	—	—	—	—	HSET	—	—	—
Tracciamento linee su plotter	—	—	—	—	—	LINE RLINE	—	—	—
Specifiche operative per il plotter	—	—	—	—	—	MODE	—	—	—
Alza la penna del plotter e la sposta in posizione specificata	—	—	—	—	—	MOVE RMOVE	— —	— —	— —
Alza la penna del plotter e la sposta all'origine	—	—	—	—	—	PHOME	—	—	—
Imposta il numero di righe per pagina	—	—	—	—	—	PAGE	—	—	—
Utilizzo del plotter come unità video	—	—	—	—	—	PLOT	—	—	—
Imposta il colore della penna del plotter	—	—	—	—	—	PCOLOR	—	—	—
Trascinamento della carta del plotter	—	—	—	—	—	SKIP	—	—	—
Test di funzionamento del plotter	—	—	—	—	—	TEST	—	—	—

APPENDICE C

COMANDI DEL D.O.S. APPLE riguardanti la gestione dei file

In Applesoft non esistono comandi BASIC per la gestione dei file. Essi sono invece inseriri nel DOS (Disk Operating System) di Apple II.

Essi possono essere eseguiti all'interno di un programma BASIC stampando, con una PRINT, una stringa che contiene un CTRL-D seguito dal comando.
Esempio:
10 D$ = CHR$ (4)
20 PRINT D$; "OPEN PIPPO"

Segue l'elenco dei comandi DOS, richiamabili da BASIC, che riguardano la gestione dei file. Viene utilizzato il formalismo sintattico di tipo COBOL, nel quale utilizzeremo:
f per nome file
d per numero dell'unità a disco (1 ÷ 2)
s per numero slot (1 ÷ 7)
v per numero volume (1 ÷ 254)
j per numero byte (1 ÷ 32767)

Le parentesi quadre racchiudono opzioni facoltative, mentre le lettere maiuscole sono simboli terminali (vanno cioè scritte come indicato)

OPEN - Serve per aprire un file

OPEN f [, Ss] [, Dd] [, Vv] per file sequenziali

OPEN f, Lj [, Ss] [, Dd] [, Vv] per file ad accesso casuale

CLOSE - Serve per chiudere un file

CLOSE [f]

WRITE - Predispone un file per la scrittura. La scrittura vera e propria dei dati avviene con delle normali PRINT.

WRITE f per file sequenziali

WRITE f [, Rr] per file ad accesso casuale, dove r è il numero progressivo del record che si vuole scrivere sul file

READ - Predispone un file per la lettura. La effettiva lettura dei dati avviene subito dopo con delle normali INPUT

READ f per file sequenziali

READ f [, Rr] per file ad accesso casuale, dove r è il numero progressivo del record che si vuole leggere dal file

APPEND - Consente di aggiungere dati al termine di un file di testo sequenziale

APPEND f [, Ss] [, Dd] [, Vv]

POSITION - Consente di accedere alle informazioni da qualsiasi campo specificato all'interno di un file di testo sequenziale

POSITION f [, Rp] dove Rp è la posizione relativa del campo

BIBLIOGRAFIA

Per TEXAS TI 99/4A

1) MANUALE D'USO a cura della TEXAS INSTRUMENTS (in dotazione all'acquisto dell'elaboratore)

PER COMMODORE VIC 20

2) VIC 20 - Per giocare con un vero computer - Manuale d'uso a cura della COMMODORE COMPUTER (in dotazione all'acquisto dell'elaboratore)
3) J. Heilborn - R. Talbott - VIC 20 USER GUIDE - Osborne/McGraw - Hill - 1983
4) R. Bonelli - IMPARIAMO A PROGRAMMARE IN BASIC con il VIC/CBM 1981 - Gruppo Editoriale Jackson
5) R. Bonelli - D. Gianni - ALLA SCOPERTA DEL VIC 20 - Gruppo Editoriale Jackson - 1983

PER COMMODORE CBM 64

6) Commodore 64 Micro Computer User Manual - Manuale d'uso a cura della COMMODORE COMPUTER (in dotazione all'acquisto dell'elaboratore)
7) COMMODORE 64 - Guida di riferimento per il programmatore - a cura della COMMODORE COMPUTER - 1983

PER SINCLAIR ZX 81

8) R. Bonelli - Guida al SINCLAIR ZX 81 e ZX 80 - Nuova Rom - Gruppo Editoriale Jackson - 1981

PER SINCLAIR ZX SPECTRUM

9) R. Bonelli - ALLA SCOPERTA DELLO ZX SPECTRUM - Gruppo Editoriale Jackson - 1983
10) DR. IAN LOGAN - IL libro del MICRODRIVE SPECTRUM - Edizioni JCE - 1984

PER SHARP MZ 700

11) MANUALE DELL'UTENTE - a cura della SHARP CORPORATION - 1983 (in dotazione all'acquisto dell'elaboratore)

PER I.B.M. P.C.

BASIC - a cura della IBM - (in dotazione all'acquisto dell'elaboratore)

PER OLIVETTI M 20

13) M 20: Introduzione al sistema: a cura della Ing. C. Olivetti & C. - 1982
14) M 20: Linguaggio BASIC - Manuale generale - a cura della Ing. C. Olivetti & C. - 1982 (in dotazione all'acquisto dell'elaboratore)
15) POCKET BASIC & PCOS x M 20 - a cura della Ing. C. Olivetti & C - 1982 (in dotazione all'acquisto dell'elaboratore)

VARIE

16) M. Sangiorgio - Strumenti di ausilio alla programmazione di problemi gestionali — Università Statale di Milano - Facoltà di Scienze - Gruppo di Elettronica e Cibernetica - 1975

INDICE ANALITICO

Q

R

S

Martino Sangiorgio, laureato in fisica presso l'Università Statale di Milano, ha acquisito una esperienza ultradecennale nel campo della elaborazione dei dati presso una grande azienda a livello nazionale, risolvendo, dapprima come programmatore e successivamente come esperto di sistemi, problemi di carattere gestionale su grossi elaboratori (mainframe).
Da alcuni anni si interessa anche di mini, personal e home computer, sia in fase di conoscenza E.D.P. che in fase didattica e hobbystica.

Il Basic è un linguaggio universale, e lo è diventato in modo definitivo da quando è stato adottato come linguaggio tipico del Personal Computer.
Eppure una delle sorprese più amare per chi comincia a programmare su di un calcolatore consiste nell'accorgersi che i suoi programmi non "girano", così come sono, su un calcolatore diverso da quello per cui sono stati scritti.
Se infatti la struttura fondamentale del BASIC è unica, e le istruzioni di carattere generale sono normalmente implementate in modo molto simile su tutti i calcolatori, è anche vero che alcuni gruppi di istruzioni (ad esempio quelle per la grafica) sono notevolmente diversi da un elaboratore ad un altro.
Lo scopo di questo libro è quello di presentare, in modo comparato, il BASIC dei microcalcolatori più diffusi (TI99/4A - VIC 20 - CBM 64 - ZX81 - ZX Spectrum - MZ 700 - IBM PC - M20 - APPLE) fornendo la chiave per poter procedere, in un momento successivo, alla conversione di programmi da un elaboratore ad un altro.

L. 45.000 COD 534A ISBN88-7056-158-5

GRUPPO
EDITORIALE
JACKSON

Martino Sangiorgio

il manuale del basic